



ÉCOLE NATIONALE SUPÉRIEURE DE TECHNIQUES AVANCÉES DE PARIS  
RAPPORT DE PROJET INTELLIGENCE ARTIFICIELLE

---

## Détection d'anomalies par segmentation sémantique

---

Élèves :

Valentin Collumeau  
Désiré Ouedraogo  
Oumaima Ben Yemna  
Jacques de Chevron-Villette  
Rania Ben Kmicha

Professeurs et Encadrants :

Gianni Franchi - ENSTA  
Nassim Belkhir - Thalès

## Sommaire

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Jeu de données . . . . .	2
1.2	État de l'art (rajouter les réfs bibli) . . . . .	2
1.3	Modèles sélectionnés . . . . .	3
1.4	Classes sélectionnées . . . . .	3
<b>2</b>	<b>Modèles</b>	<b>5</b>
2.1	SPADE[3] . . . . .	5
2.1.1	Protocole d'expérimentation et résultats . . . . .	5
2.1.2	Optimisation des hyperparamètres . . . . .	6
2.1.3	Avantages et limites du modèle et de l'étude . . . . .	6
2.2	PaDiM . . . . .	7
2.2.1	Protocole d'expérimentation . . . . .	7
2.2.2	Résultats . . . . .	7
2.3	PatchCore . . . . .	10
2.3.1	Protocole d'expérimentation . . . . .	10
2.3.2	Résultats . . . . .	10
2.3.3	Optimisation hyperparamètres . . . . .	12
2.4	RIAD . . . . .	14
2.4.1	Protocole d'expérimentation . . . . .	14
2.4.2	Résultats . . . . .	14
2.5	Normalizing flows-FastFlow . . . . .	17
2.5.1	Protocole d'expérimentation . . . . .	17
2.5.2	Résultats . . . . .	17
<b>3</b>	<b>Conclusion</b>	<b>21</b>
3.1	Comparaison . . . . .	21
3.1.1	Résultats . . . . .	21
3.1.2	Analyse . . . . .	21
<b>4</b>	<b>Retour d'expérience</b>	<b>23</b>
<b>5</b>	<b>Annexe</b>	<b>25</b>
5.1	Annexe 1 . . . . .	25
5.2	Annexe 2 : notes techniques . . . . .	26
5.2.1	SPADE . . . . .	26
5.2.2	Description . . . . .	27
5.2.3	Description . . . . .	28
5.2.4	Description de modèle . . . . .	32

# 1 Introduction

La détection d'anomalie est un domaine de l'apprentissage automatique qui consiste à identifier des échantillons d'un ensemble de données qui ne suivent pas la même distribution que les autres échantillons. Cette tâche est importante dans de nombreux domaines, notamment dans l'industrie où elle peut être utilisée pour détecter des défauts sur des produits ou des composants.

## 1.1 Jeu de données

Le challenge MVTEC est une compétition de détection d'anomalie organisée chaque année depuis 2019. Le but de ce challenge est d'évaluer les performances des différents modèles de détection d'anomalie sur des images de produits manufacturés avec des défauts tels que des rayures, des fissures, des bulles, etc.

Pour notre projet de détection d'anomalies, nous avons choisi d'utiliser l'ensemble de données MVTEC AD. Cet ensemble de données contient 15 classes d'anomalies différentes, telles que des rayures, des fissures, des bulles, des inclusions, des déformations, des trous, etc. Chaque classe contient entre 80 et 250 images de produits normaux et de produits anormaux, pour un total d'environ 5000 images (cf Fig.1).

Les images de l'ensemble de données MVTEC AD sont fournies sous forme de fichiers PNG de taille variant de 700x700 à 1024x1024. Les images normales sont des images de produits manufacturés sans défauts, tandis que les images anormales contiennent un défaut ou une anomalie qui doit être détecté par le modèle de détection d'anomalies. Le ground truth fourni avec chaque image indique les pixels où se trouve l'anomalie.

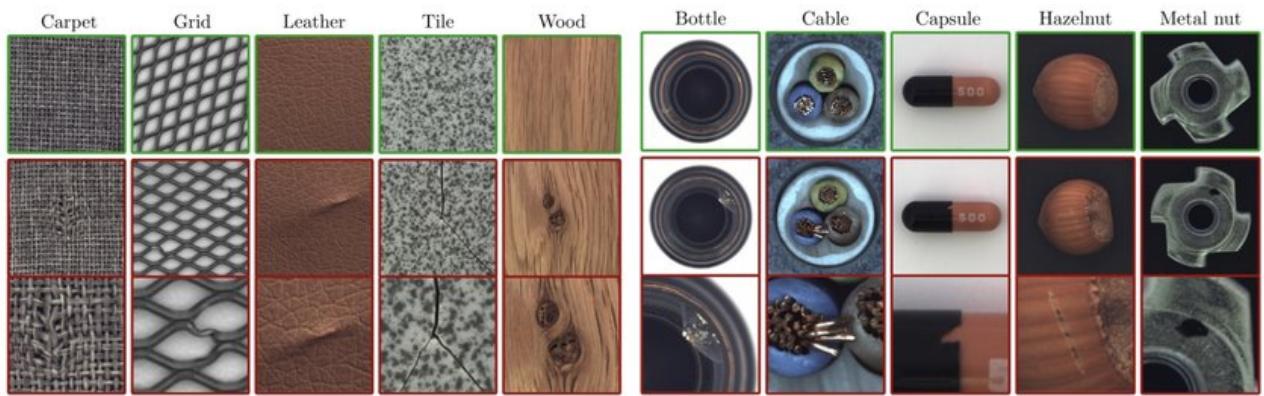


FIGURE 1 – Extrait de 10 classes du dataset MVTec AD utilisé

## 1.2 État de l'art (rajouter les réfs bibli)

À l'heure actuelle, l'état de l'art des meilleurs algorithmes sur le MVTec Challenge utilise souvent des architectures de réseaux de neurones profonds, telles que les réseaux de neurones convolutionnels (CNN) ou les réseaux de neurones avec des flots normalisants (NFF). Certains des modèles les plus performants incluent :

- **SPADE** [3] : le premier algorithme a avoir obtenu des résultats satisfaisants. À partir de la comparaison de caractéristiques extraites d'un modèle entre l'image test et les autres, celui-ci identifie et segmente les anomalies. Il sera décrit plus bas.
- **PaDiM** [5] : une approche basée sur des patchs qui utilise une distance Mahalanobis pour calculer l'anomalie de chaque patch et agrège les résultats pour obtenir une carte globale de l'anomalie. PaDiM a remporté la première place de l'ensemble de données MVTec AD en 2019. *Ce modèle a été étudié dans notre travail, ainsi qu'un autre similaire PatchCORE [4].*
- **GANomaly** : un modèle basé sur des réseaux de neurones génératifs antagonistes (GAN) qui apprend une distribution des images normales et détecte les anomalies en utilisant la différence entre la reconstruction de l'image originale et la reconstruction générée par le générateur du GAN. GANomaly a obtenu des performances de pointe sur l'ensemble de données MVTec AD.

- **AE-IF** : un modèle qui utilise un auto-encodeur (AE) pour apprendre une représentation latente des images normales, puis calcule l'anomalie en utilisant la distance de Mahalanobis entre les représentations latentes des images normales et anormales. AE-IF a également obtenu des performances de pointe sur l'ensemble de données MVTEC AD. *Un modèle proche appelé RIAD a été testé dans notre travail.*

Il est important de noter que les performances des différents modèles peuvent varier en fonction des caractéristiques des données et des types d'anomalies à détecter. C'est pour cela que nous avons décidé de tester plusieurs modèles à tester sur des classes différentes.

### 1.3 Modèles sélectionnés

Plusieurs modèles ont été proposés pour résoudre ce problème. Le modèle **SPADE** (Spatially-Adaptive De-normalization) utilise un réseau de neurones convolutionnel pour prédire une carte de normalisation qui est ensuite utilisée pour reconstruire l'image originale. Le modèle **PATCHCORE** utilise également un réseau de neurones convolutionnel, mais avec une architecture en blocs pour capturer des motifs à différentes échelles. Le modèle **PADIM** (Patch-based Anomaly Detection via Interpretable Multi-resolution) utilise une approche basée sur des patchs pour détecter les anomalies à différentes échelles. Enfin, le modèle **RIAD** (Reconstruction-based Anomaly Detection) utilise une technique d'autoencoder masqué pour reconstruire des images parfaites, et les comparés aux réelles images pour identifier les potentiels défauts.

Une méthode de "**Normalizing Flow**" a aussi été testé. Peu de travaux utilisent cette méthode sur le challenge MVTEC, mais les perspectives de résultats sont importantes.

Chacun de ces modèles a ses avantages et ses limites en fonction des caractéristiques des données et des types d'anomalies à détecter. Il est important de tester plusieurs modèles pour trouver celui qui convient le mieux à chaque ensemble de données et au problème de détection d'anomalie spécifique.

### 1.4 Classes sélectionnées

Dans l'ensemble de données MVTEC AD, les 15 classes d'anomalies sont divisées en deux catégories : les textures et les objets. Le détail complet de ces classes est présent ci-dessous (cf Fig 2).

Les classes de textures contiennent des images d'objets manufacturés avec des défauts de texture tels que des rayures, des taches, des marques de doigts, des bosses, etc. Ces défauts de texture peuvent être subtils et difficiles à détecter pour un modèle de détection d'anomalies. Les classes de textures incluent des anomalies telles que "Carpet", "Leather", "Tile", "Wood", "Metal", "Transistor" et "Screw".

Les classes d'objets contiennent des images d'objets manufacturés avec des défauts structurels tels que des fissures, des trous, des inclusions, des déformations, etc. Ces défauts de structure sont souvent plus évidents que les défauts de texture et plus faciles à détecter pour un modèle de détection d'anomalies. Les classes d'objets incluent des anomalies telles que "Bottle", "Cable", "Capsule", "Hazelnut", "Pill", "Toothbrush", "Metal Nut" et "Zipper".

	Category	# Train	# Test (good)	# Test (defective)	# Defect groups	# Defect regions	Image side length
Textures	Carpet	280	28	89	5	97	1024
	Grid	264	21	57	5	170	1024
	Leather	245	32	92	5	99	1024
	Tile	230	33	84	5	86	840
	Wood	247	19	60	5	168	1024
Objects	Bottle	209	20	63	3	68	900
	Cable	224	58	92	8	151	1024
	Capsule	219	23	109	5	114	1000
	Hazelnut	391	40	70	4	136	1024
	Metal Nut	220	22	93	4	132	700
	Pill	267	26	141	7	245	800
	Screw	320	41	119	5	135	1024
	Toothbrush	60	12	30	1	66	1024
	Transistor	213	60	40	4	44	1024
	Zipper	240	32	119	7	177	1024
Total		3629	467	1258	73	1888	-

FIGURE 2 – Détail des classes du dataset

Il est important de tester les deux types de classes car les défauts de texture et les défauts structurels sont souvent détectés différemment par les modèles de détection d'anomalies. Les défauts de texture sont souvent plus subtils et nécessitent des modèles de détection d'anomalies plus sophistiqués. D'un autre côté, les défauts structurels sont souvent plus évidents et peuvent être détectés par des modèles de détection d'anomalies plus simples. Tester les deux types de classes permet de s'assurer que le modèle de détection d'anomalies est robuste et performant pour détecter différents types d'anomalies, ce qui peut être important dans un contexte industriel où différents types d'anomalies peuvent être présents. Ainsi, nous avons décidé de tester, optimiser et comparer les différents modèles selon les trois classes suivantes :

- "wood" pour tester sur les textures ;
- "screw" pour tester les objets contenant des rotations ;
- "bottle" qui peut être une classe complexe.

Les résultats sur chaque classe seront aussi affichés mais sans certitudes d'être optimaux.

---

## 2 Modèles

### 2.1 SPADE[3]

Le modèle SPADE est le modèle affiché comme le premier ayant obtenu des résultats acceptables sur le jeu de données MvTec.

#### 2.1.1 Protocole d'expérimentation et résultats

Pour vérifier les résultats qu'obtient cet algorithme sur la segmentation d'anomalies du jeu de données MvTec, un protocole assez succinct est mis en place.

L'algorithme SPADE ne nécessitant aucun entraînement, ce protocole de test se résume à l'implémentation des différentes étapes décrites dans la section précédente.

L'extraction des caractéristiques est faite pour l'ensemble des données *train* et *test*.

La distance euclidienne entre les caractéristiques de chaque image de test et celles de chaque image d'entraînement du jeu de données est ensuite calculée.

Ensuite, les images de test sont comparées avec les caractéristiques "moyennes" construites à partir des  $k$ -plus proches images de l'ensemble d'entraînement, d'abord à l'échelle de l'image puis à l'échelle du pixel. Pour des raisons de simplicité et de temps de calcul, cette étape a été implémentée avec  $k = 1$  uniquement.

Cela permet, dans un premier temps, d'obtenir la métrique ROCAUC des anomalies sur les images, puis d'extraire une liste de scores par pixel pour segmenter les anomalies à partir d'un seuillage et d'un filtre gaussien, avant de calculer le ROCAUC au niveau des pixels.

Ces étapes peuvent être répétées pour chacune des classes étudiées, et ce protocole permet d'obtenir les résultats de classification des images et des pixels.

On constate que cet algorithme est plutôt bon pour identifier les images contenant des anomalies (hormis pour la classe *screw*) mais a de grosses difficultés à segmenter ces anomalies. Ci-dessous se trouvent les ROCAUC au niveau image (à gauche) et au niveau pixel (à droite).

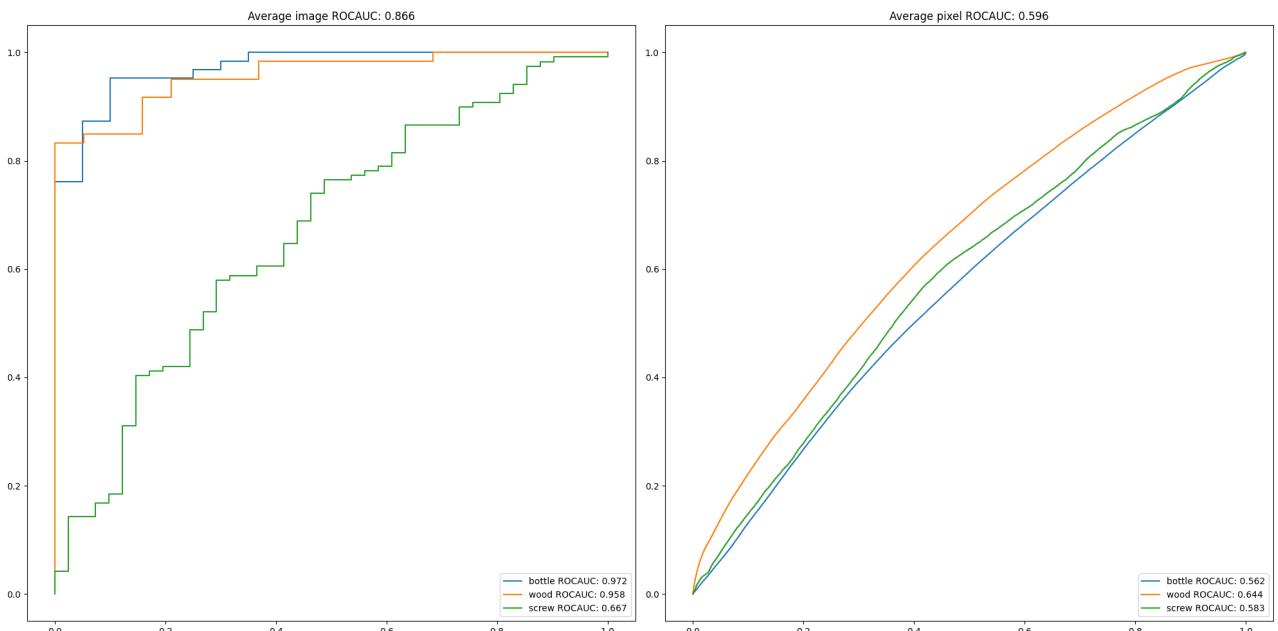


FIGURE 3 – Résultats pour  $k = 5$  voisins dans la classification des images

### 2.1.2 Optimisation des hyperparamètres

Pour chercher le  $k$  optimal, une simple recherche en grille (*gridsearch* pour les anglophones) permet de tester diverses valeurs, en comparant les résultats.

Pour divers nombre de voisins considérés, on obtient les résultats ci-dessous au niveau de la performance "images".

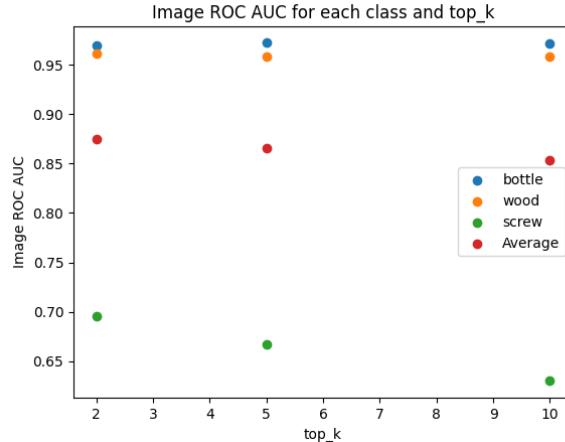


FIGURE 4 – Résultats pour divers  $k$  voisins dans la classification des images

Ces comparaisons sont faites pour la classification des images, l'implémentation n'ayant pas été faite pour la carte de scores (*scoremap*) pour  $k > 1$ .

On constate que le nombre de voisins variant entre 2 et 10 ne semble pas avoir une importance significative sur les performances de SPADE sur les deux classes *wood* et *bottle*. Néanmoins, un plus petit nombre de voisins semble bénéficier aux performances sur la classe *screw*.

### 2.1.3 Avantages et limites du modèle et de l'étude

L'énorme avantage de l'algorithme SPADE est qu'il ne nécessite aucun entraînement. Cela permet une simplicité et un temps de calcul appréciables.

Néanmoins, cet algorithme est celui qui, parmi ceux que nous avons sélectionnés, semble le moins performant sur la tâche de segmentation d'anomalies.

Une des grandes limites de cette étude de SPADE est qu'elle a été restreinte à une carte de caractéristique d'un unique plus proche voisin. Pour la compléter, étudier et rechercher le meilleur  $k$  pour construire la carte de score à partir des  $k$  plus proches voisins permettrait, au-delà de classifier une image, de segmenter les anomalies éventuellement contenues dans cette image. Cela n'a pas été fait par manque de temps et, surtout, de temps de calcul (cf. retours d'expérience).

L'étude des hyperparamètres pourrait également être étendue à un bien plus grand nombre de voisins (par exemple  $k = 100$ ) et sur un échantillonage plus grand pour véritablement constater les différences de performances. Pour cela, une grande puissance de calcul est également nécessaire.

Enfin, pour aller encore plus loin, tester cet algorithme sur des images bruitées permettrait d'évaluer la robustesse de celui-ci, et la comparer à celles des autres modèles.

## 2.2 PaDiM

PaDiM est un modèle d'apprentissage automatique pour la détection des anomalies publié en novembre 2020 [5]. Il utilise un réseau neuronal convolutionnel (CNN) pré-entraîné pour l'extraction de caractéristiques et possède les deux propriétés suivantes :

- Chaque position de patch est décrite par une distribution gaussienne multivariée.
- PaDiM prend en compte les corrélations entre les différents niveaux sémantiques d'un CNN pré-entraîné.

### 2.2.1 Protocole d'expérimentation

Dans le but d'obtenir des résultats, nous avons commencé par tester notre modèle sur toutes les classes, puis nous avons sélectionné les meilleures et les mauvaises classes. Nous avons ensuite étudié les performances du PaDiM en fonction de différents hyperparamètres tels que l'architecture du backbone et le choix des couches d'extraction (couche 1, couche 2 ou couche 3). Enfin, nous avons effectué l'optimisation du modèle sur les classes sélectionnées.

### 2.2.2 Résultats

Nous avons d'abord testé notre modèle sur toutes les classes en prenant ResNet18 comme architecture backbone et l'extraction des paramètres se fait sur les 3 premières couches du réseau CNN pré-entraîné. La figure suivante représente les résultats obtenus.

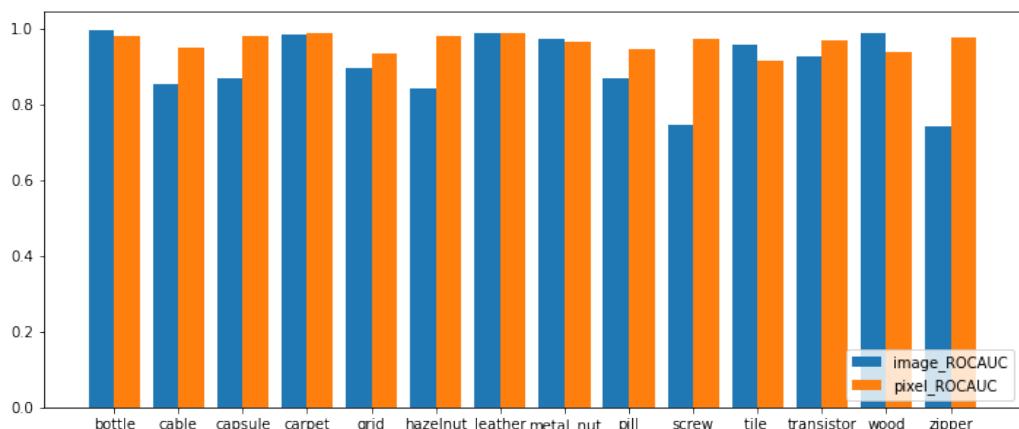


FIGURE 5 – Les résultats du modèle PaDiM

La figure 5 montre que les performances de notre modèle varient en fonction des différentes classes. Par exemple, les classes "bottle", "carpet" et "wood" ont une valeur de ROCAUC d'image supérieure à 0,98 alors que les classes "zipper" et "screw" ont une valeur qui ne dépasse pas 0,7. Il est donc important d'étudier l'influence de différents hyperparamètres sur les prédictions de notre modèle PaDiM.

### L'architecture du modèle de base :

Le modèle PaDiM utilise des modèles de base pour extraire des caractéristiques à partir des images d'entrée. Ces caractéristiques sont ensuite utilisées pour apprendre la distribution des tâches dans les images normales et détecter les anomalies dans les images d'entrée. Le choix du modèle de base est important car il doit être capable d'extraire des caractéristiques discriminantes pertinentes pour la tâche de détection des anomalies. Le modèle doit être suffisamment profond et complexe pour extraire des caractéristiques de haut niveau, mais aussi suffisamment léger pour permettre un apprentissage efficace sur de grandes quantités de données.

Les modèles de base ResNet18 et WideResNet50 sont deux choix courants pour l'extraction de caractéristiques dans le modèle PaDiM en raison de leurs performances éprouvées dans les tâches de classification d'images.

ResNet18 est un réseau de convolution relativement léger avec 18 couches et WideResNet50 est une version plus complexe de ResNet avec 50 couches et des blocs de convolution plus grands.

Le tableau 1 représente les résultats obtenus par le modèle PaDiM en utilisant WideResnet50 comme modèle de base. Nous pouvons constater des améliorations significatives des valeurs ROCAUC pour les différentes classes sélectionnées. Par exemple, la classe " screw " atteint une valeur de 0,844 avec WideResnet50. Cela peut s'expliquer par le fait que ce modèle est plus profond que ResNet18 et utilise des blocs de convolution plus grands, ce qui signifie que chaque couche de convolution contient plus de filtres. Cela permet au réseau de capturer plus de détails dans les images et de mieux représenter les caractéristiques visuelles, ce qui améliore les performances de classification.

Classe	PaDiM ResNet18		PaDiM WideResnet50	
	Pixel ROCAUC	Image ROCAUC	Pixel ROCAUC	Image ROCAUC
bottle	0.981	0.996	0.982	0.998
zipper	0.976	0.741	0.984	0.909
wood	0.940	0.99	0.941	0.988
screw	0.972	0.745	0.983	0.844
carpet	0.988	0.984	0.990	0.999
tile	0.917	0.959	0.939	0.974

TABLE 1 – Les résultats du modèle PaDiM.

Les figures 6 et 7 nous permettent de comparer visuellement les résultats obtenus avec les deux modèles pour la même image d'entrée. Nous constatons que WideResNet50 est capable de segmenter les anomalies de l'image d'entrée avec plus de précision que l'architecture ResNet18.

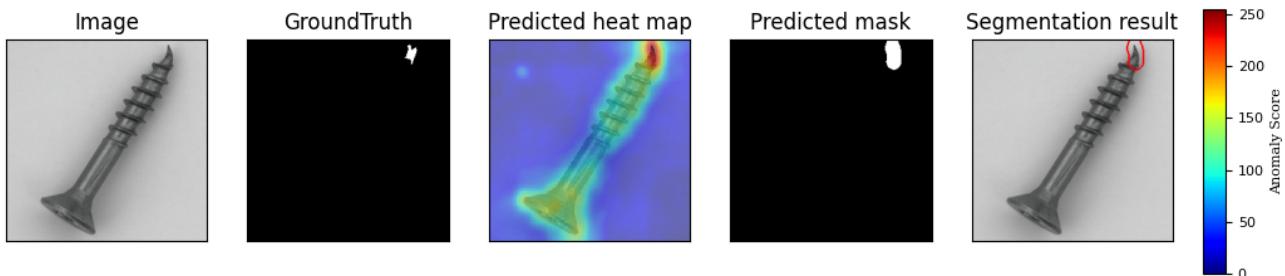


FIGURE 6 – Exemple de prédiction du modèle PaDiM en utilisant ResNet18.

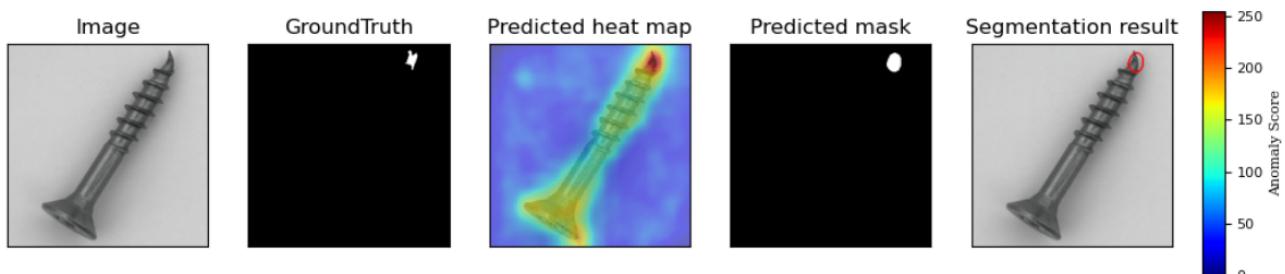


FIGURE 7 – Exemple de prédiction du modèle PaDiM en utilisant WideResNet50.

#### – Comparaison avec les données de référence :

Le tableau 9 dans l'annexe 1, représente les résultats fournis dans l'article "Patch Distribution Modeling for Anomaly Detection and Localization"[5]. Nous pouvons confirmer que le modèle WideResnet50 donne les meilleurs résultats pour les différentes classes et nous constatons que nos résultats sont très proches des valeurs mentionnées dans l'article.

### Les couches d'extraction :

Dans le modèle PADIM, le choix des couches d'extraction des caractéristiques est crucial car il peut avoir un impact significatif sur la performance de la détection des anomalies. Les couches d'extraction de caractéristiques sont chargées de transformer les images d'entrée en vecteurs de caractéristiques qui sont ensuite utilisés pour la détection d'anomalies. Il est important de choisir des couches qui extraient des caractéristiques discriminantes, c'est-à-dire des caractéristiques qui permettent de distinguer les images normales des images anormales.

Nous avons donc étudié les effets de la sélection des couches d'extraction de caractéristiques sur notre modèle et nous avons obtenu les résultats présentés dans le tableau 2.

Les couches	ROCAUC image sur toutes les classes
Couche 1	0.807
Couche 2	0.81
Couche 3	0.86
Couche 1+2+3	0.88

TABLE 2 – Les performances du PaDiM en variant les couches d'extraction.

Le tableau 2 montre que les meilleurs résultats sont obtenus lorsque les trois premières couches du modèle ResNet18 sont utilisées pour l'extraction des caractéristiques de l'image d'entrée, alors que les résultats sont moins bons lorsqu'une seule couche est utilisée. On constate également que les valeurs ROCAUC obtenues sont meilleures lorsque seule la troisième couche est utilisée par rapport à la deuxième et à la première couche. Cela peut s'expliquer par le fait que la troisième couche contient un niveau d'information sémantique plus élevé et peut mieux décrire les caractéristiques.

#### – Comparaison avec les données de référence :

Les résultats de référence sont présentés dans le tableau 10 dans l'annexe 1, dans lequel nous constatons que les meilleurs résultats sont obtenus lors de l'utilisation des trois premières couches pour l'extraction des caractéristiques avec l'architecture Resnet18. Nous remarquons également que les valeurs issues de l'article sont meilleures que les valeurs obtenues, ce qui peut s'expliquer par le choix de paramètres susceptibles d'influencer les performances du modèle, comme le nombre de dimensions à sélectionner aléatoirement dans les vecteurs de caractéristiques intermédiaire.

### Test sur données bruitées :

Pour étudier la robustesse de notre modèle Padim WideResNet50, nous avons ajouté du bruit aux données de test des classes wood, screw et bottle et nous avons obtenu les résultats présentés dans le tableau 3. Nous pouvons remarquer que les performances de notre modèle diminuent avec les données bruitées. Par exemple, pour la classe screw, la valeur ROCAUC de l'image ne dépasse pas 0,7, alors qu'elle est de 0,84 pour les données non bruitées.

Classe	PaDiM WideResnet50	
	Image ROCAUC	Pixel ROCAUC
bottle	0.92	0.90
screw	0.702	0.91
wood	0.95	0.905

TABLE 3 – Les résultats du modèle PaDiM avec les données bruitées.

## 2.3 PatchCore

### 2.3.1 Protocole d'expérimentation

Dans un premier temps, nous avons essayé de reproduire les résultats du papier [4]. Cela se fait en testant sur toutes les classes de la base de donnée MVTec AD en respectant les hyperparamètres utilisés. Ces hyperparamètres sont :

- Paramètres spécifiques au pré-entraînement :

**backbone**

**couches d'extraction**

- Paramètres de l'évaluateur d'anomalies du plus proche voisin :

**nombre des plus proches voisins**

- Paramètres de patch :

**taille de patch**

**chevauchement de patch**

- Paramètres de l'échantillonneur :

**nom de l'échantillonneur**

**pourcentage d'échantillonnage**

Par la suite, plusieurs autres configurations d'hyperparamètres seront à tester et les résultats obtenus expliqués.

### 2.3.2 Résultats

Avec un backbone wideresnet50, les couches d'extraction 2 et 3, la taille de patch 3 sans chevauchement, le nombre des plus proches voisins à 1 et l'échantillonner Greedy Coreset à 1%, les résultats sont les suivants sur tous le dataset MVTec AD 4 :

Model	Mean AUROC	Mean Seg. AUROC
WR50-baseline	99.2%	98.1%

TABLE 4 – Les résultats de référence du modèle PatchCore sur tout le dataset.

Nous avons entrepris de tester individuellement chaque classe de la base de donnée et avons obtenu les résultats du tableau 5.

Row Names	instance_auroc	full_pixel_auroc	anomaly_pixel_auroc
bottle	1	0.9848388245	0.9795612141
capsule	0.9792580774	0.9895443665	0.9874208704
cable	0.996814093	0.9841371899	0.9752826803
wood	0.9912280702	0.9507743423	0.9373950758
leather	1	0.992881083	0.9903823092
screw	0.987702398	0.9952476919	0.9938280112
grid	0.9791144528	0.988007846	0.9836616425
transistor	0.99875	0.960981119	0.9274338821
zipper	0.9950105042	0.9886782459	0.9856368239
pill	0.966721222	0.9780095292	0.9760475848
hazelnut	1	0.9867676801	0.9791327108
tile	0.9949494949	0.9571209382	0.9407940304
carpet	0.9859550562	0.9905142002	0.987865855
toothbrush	1	0.9857315492	0.9799756312
metal_nut	0.9990224829	0.9834029347	0.9792238336

TABLE 5 – Les résultats de PatchCore sur chaque classe.

Les images de segmentations des classes screw, wood et bottle sont présentées dans les figures 8, 9 et 10. Les résultats sont très clairement corrects et les zones de défauts sont repérés avec exactitude.

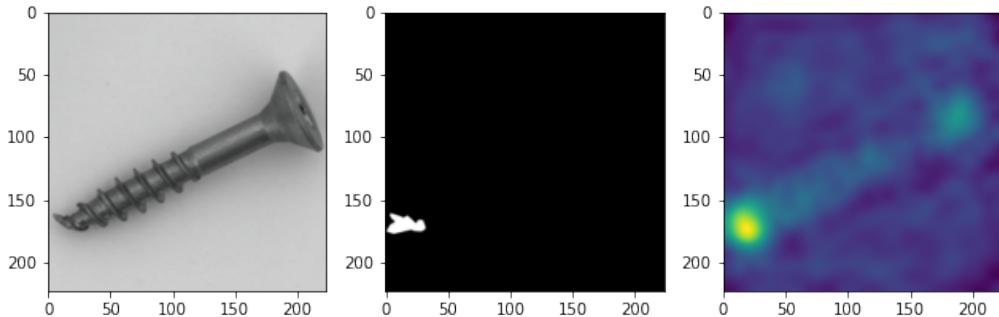


FIGURE 8 – Segmentation de la classe screw avec PatchCore

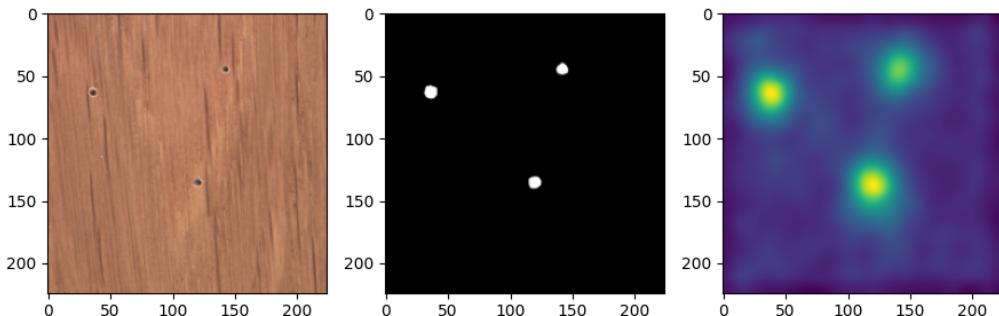


FIGURE 9 – Segmentation de la classe wood avec PatchCore

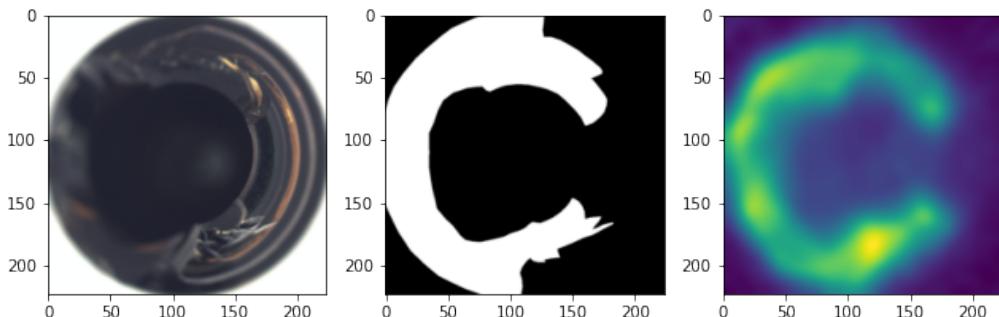


FIGURE 10 – Segmentation de la classe bottle avec PatchCore

#### Test sur données bruitées

Nous avons appliquer aléatoirement (probabilité de 0.3) du bruit de flottage sur les données de test des classes wood, screw et bottle. Les transformations ainsi obtenues sont présentées dans les figures 13. Ensuite, nous avons comparé les performances du modèle. Les résultats sont visibles dans la figure 14. On note bien une baisse de performance notamment des scores 'full\_pixel\_auroc' et 'anomaly\_pixel\_auroc'.

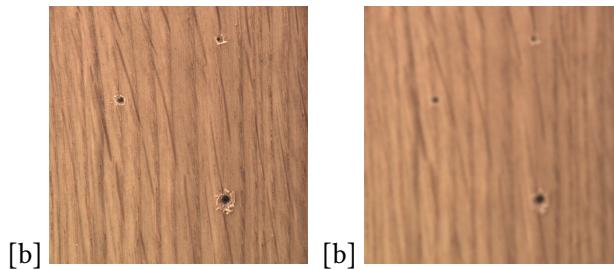


FIGURE 11 – Exemple de bruits sur la classe wood

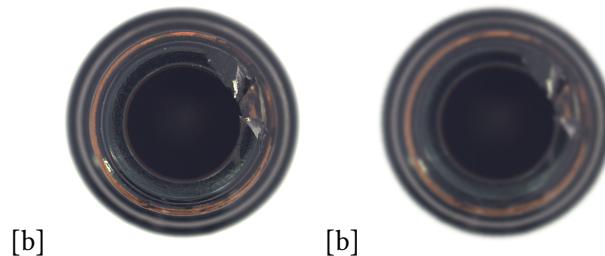


FIGURE 12 – Exemple de bruits sur la classe bottle

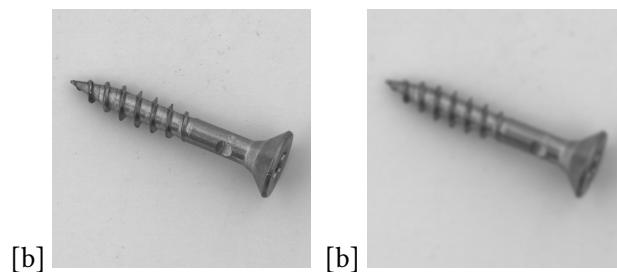


FIGURE 13 – Exemple de bruits sur la classe screw

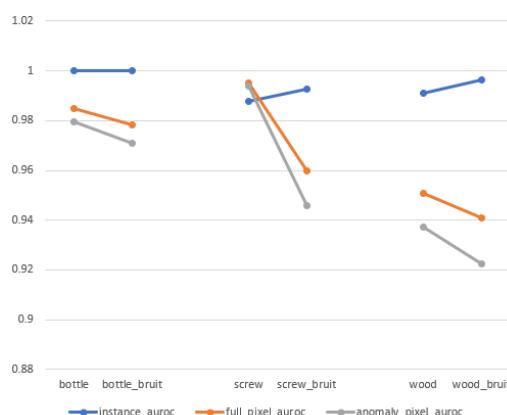


FIGURE 14 – Performances sur les classes buités avec PatchCore

### 2.3.3 Optimisation hyperparamètres

Nous avons fait varier certains hyperparamètres pour voir leur impact sur les performances du modèle. Tout d'abord, nous avons fait plusieurs tests en gardant fixés tous les hyperparamètres de base tout en faisant varier la taille de patch. Les résultats sur la classe wood sont présentés dans la figure 15. On remarque que lorsqu'on augmente la taille de patch, les performances au niveau des pixels baissent ce qui est un comportement attendu.

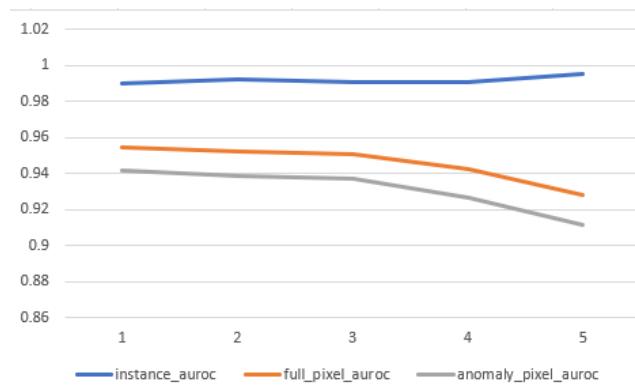


FIGURE 15 – Evolution selon taille de patch sur la classe wood avec PatchCore

Egalement, différents niveaux de sous échantillonnage ont été testés avec le CoreSet sampling. Le résultats sont visibles dans la figure 16. On observe une baisse très légère de performance au fur et à mesure qu'on augmente le pourcentage de sous échantillonage.

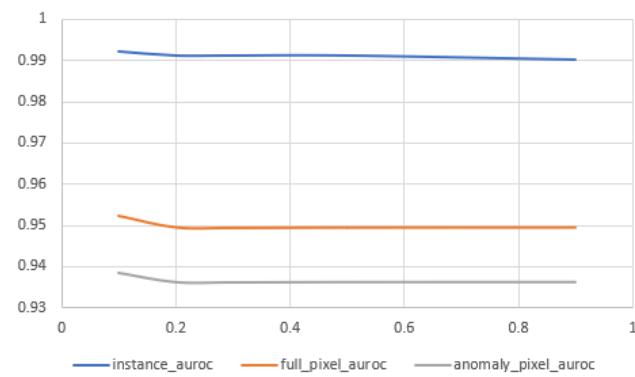


FIGURE 16 – Evolution selon le pourcentage de sous échantillonnage sur la classe wood avec PatchCore

## 2.4 RIAD

Le modèle RIAD - Reconstruction by Inpainting for visual Anomaly Detection - se base sur la technique des mask autoencoders (cf figure 33). Il est très intéressant lorsqu'il faut identifier une zone qui fait dévier l'apparence habituelle d'un objet

### 2.4.1 Protocole d'expérimentation

Le protocole d'expérimentation mis en place pour le test et l'optimisation de ce modèle a été relativement simple. L'enchaînement suivant d'étapes a été suivi :

- tester le modèle avec des hyperparamètres simples sur toutes les classes ;
- comparer sur toutes les classes avec les résultats de l'état de l'art pour voir si l'ordre de grandeur des résultats obtenus était cohérent ;
- faire quelques essais en modifiant les hyperparamètres classiques - taille d'images, batch size - pour en déduire une influence générale sur toutes les classes, et étudier la robustesse du modèle face à ces hyperparamètres simples ;
- le reste de l'optimisation a été réalisé sur les classes "bottle", "wood" et "screw" comme pour le reste des modèles - cf partie classes -.

### 2.4.2 Résultats

Les premiers résultats sont présentés ci-dessous (cf Fig 17). Chaque étape de l'algorithme a été affiché pour permettre de repérer facilement la partie défaillante lorsque le modèle marche moins bien. Les défauts sont visibles facilement sur les deux exemples ci dessous. Le choix du seuil pour définir si oui ou non il y a un défaut est très important. Le plus simple est de tracer une ROC curve ou une Precision/Recall curve et de définir le seuil en fonction du point que l'on souhaite obtenir.

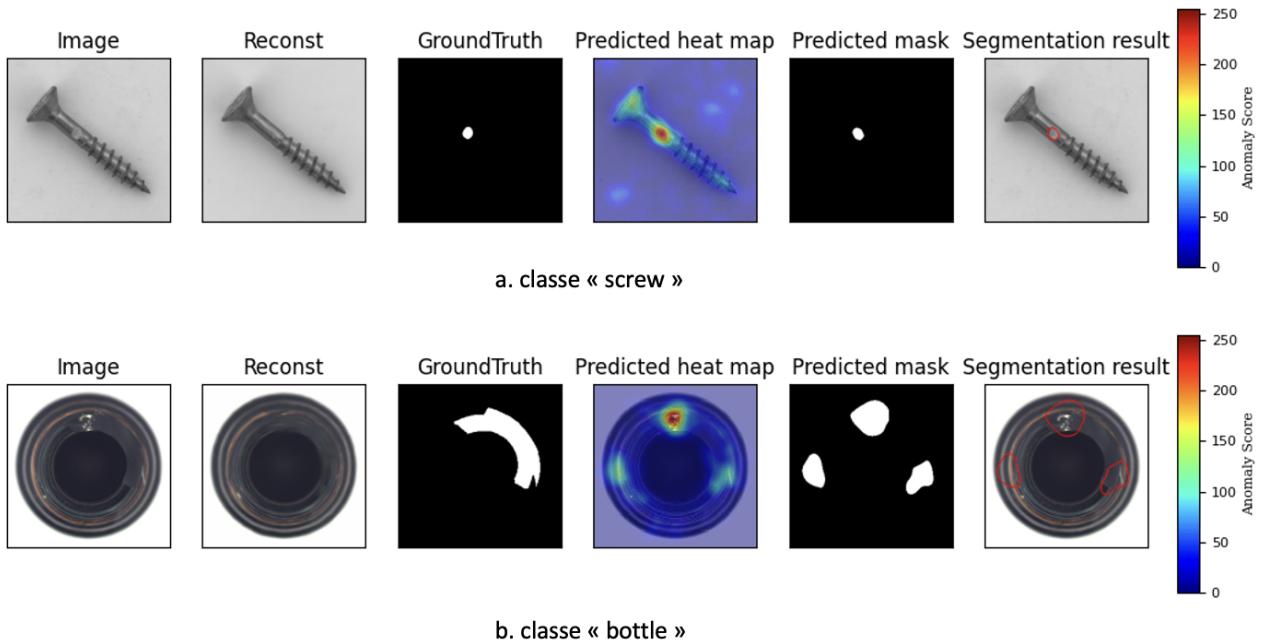


FIGURE 17 – Segmentation des classes "bottle" et "screw"

Le tableau ci dessous représente les scores pour chaque classe de l'algorithme (cf Tab. 6 & 18). On remarque que pour certaines classes, il y a de fortes disparités entre le score évalué sur l'image ou celui sur les pixels : screw, pill, capsule par exemple. Le score par image est un peu moins représentatif car il y a moins d'images que de pixels, mais une telle différence est quand même étonnante.

Row Names	anomaly_image_auroc	anomaly_pixel_auroc
bottle	0.946	0.985
capsule	0.978	0.748
cable	0.778	0.764
wood	0.796	0.854
leather	0.996	0.971
screw	0.983	0.564
grid	0.986	0.950
transistor	0.806	0.844
pill	0.960	0.660
hazelnut	0.970	0.803
tile	0.813	0.802
carpet	0.909	0.627
toothbrush	0.986	0.983
metal_nut	0.893	0.695

TABLE 6 – Les résultats de PatchCore sur chaque classe.

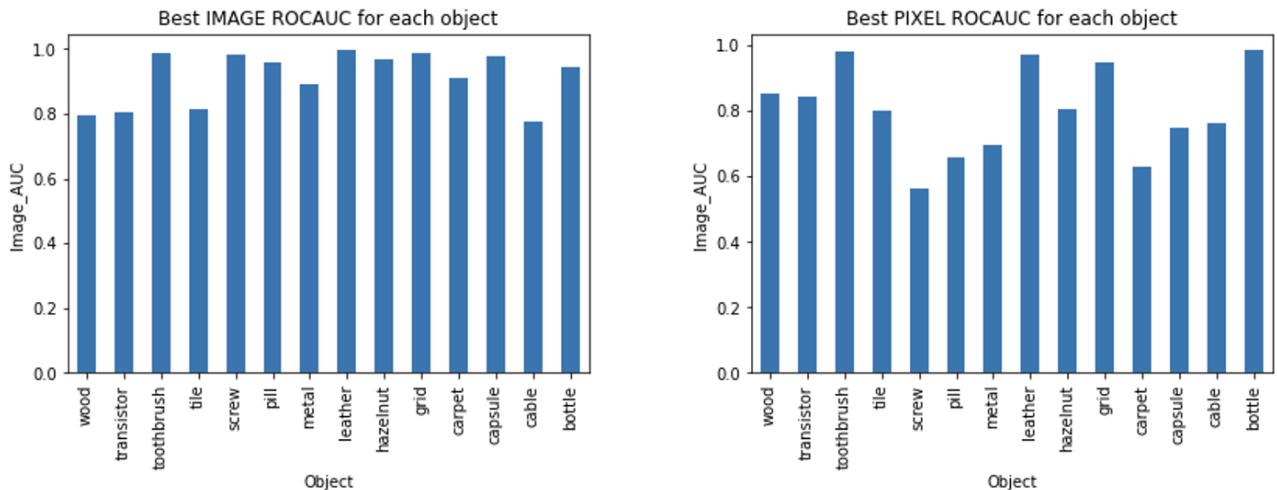


FIGURE 18 – Graphique des résultats de RIAD classe par classe

Plusieurs expériences ont été réalisées pour essayer d'identifier les meilleurs hyperparamètres. La figure suivante montre l'étude de deux d'entre eux : la taille de l'image en entrée du réseau, et la taille des batchs d'entraînements (cf Fig 19). Ces essais ont été réalisés sur plusieurs classes, mais les images ci-dessous représentent de la classe "screw".

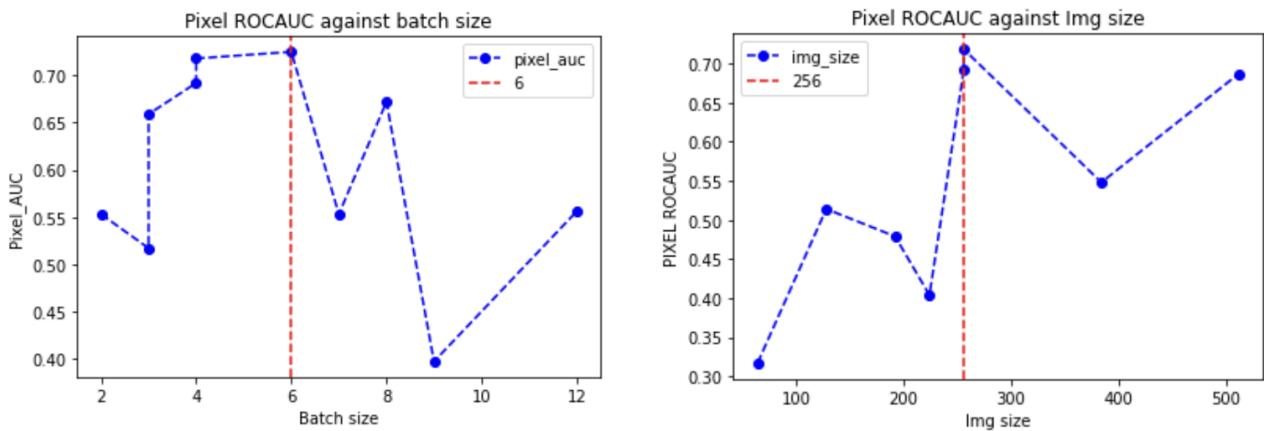


FIGURE 19 – Résultats des tests des hyperparamètres : "taille des batchs" et "taille des images" sur la classe screw.

La taille des images la plus intéressante est de loin la taille 256. L'augmenter n'apporte pas forcément de meilleures performances, alors que le temps de calcul augmente lui considérablement. Du coup de taille des batchs, c'est 6 qui apparaît le plus intéressant, même si on remarque qu'il a moins d'importances que la taille des images. D'autres hyperparamètres ont été testés, comme la taille des grilles où le pourcentage d'image caché.

Ce modèle a l'avantage d'être facilement interprétable, et très précis. En revanche, la quantité de données nécessaire à un bon entraînement est importante, tout comme son temps de calcul. Il ne s'adapte pas à des tâches plus générales ou à de l'étude de vidéos par exemple.

## 2.5 Normalizing flows-FastFlow

La technique Normalizing Flow était connue pour la génération des images au fil des années. Par conséquent, elle représente un défi continu pour son utilisation dans le domaine de détection d'anomalie.

FastFlow [2] est un algorithme de détection et localisation d'anomalie par un apprentissage non supervisé. Ce modèle se base essentiellement sur l'utilisation de la technique Normalizing FLOW [1] c'est à dire l'application des transformations inversibles sur la distribution des caractéristiques initiale dans le but d'obtenir une distribution simple(normale).

FastFlow se compose des étapes suivants détaillés en Annexe :

- Extraction des caractéristiques
- Estimation de la distribution d'entrée
- Détection et localisations des anomalies avec des différentes distributions

### 2.5.1 Protocole d'expérimentation

En premier lieu, on teste le modèle FastFlow avec les hyperparamètres de l'état de l'art sur toutes les classes et en compare les résultats obtenus avec ceux en théorique. En second lieu, on commence à changer certains hyperparamètres comme le Batch Size ,learning rate, index block pour visualiser la performance générale de notre modèle sur les classes. En dernier lieu, suite à un choix commun des classes, on optimise les hyperparamètres et on visualise les résultats selon cette sélection.

#### Hyperparamètres

Ce modèle présente plusieurs hyperparamètres qui jouent un rôle crucial sur sa performance. On y distingue :

- La taille initiale de l'image d'entrée.
- La taille des caractéristiques dans les couches des réseaux de neurones
- Nature de couche de convolution dans le FastFlow( $3 \times 3$  ou  $1 \times 1$ )
- Indice des blocks de réseau de neurones desquels on peut extraire les caractéristiques discriminatives.
- Nombre de step pour FastFlow.
- Batch size
- Learning rate pour l'algorithme d'optimisation Adam.
- Nombre d'époques pour l'entraînement.

### 2.5.2 Résultats

Les figures ci-dessous représentent les résultats faites par le modèle FastFlow lors de l'étape de test en comparant les images normales et anormales des classes choisies "bottle", "wood" et "screw".

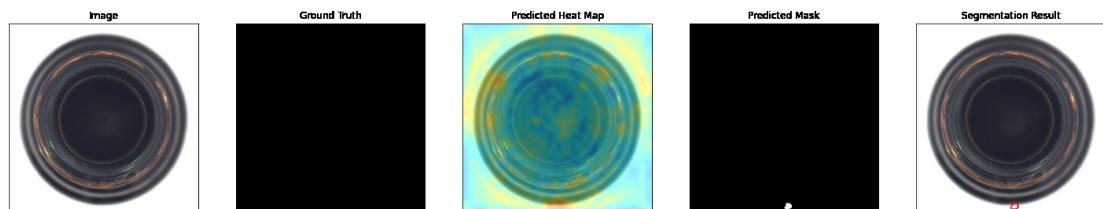


FIGURE 20 – Image normale de la classe "bottle"

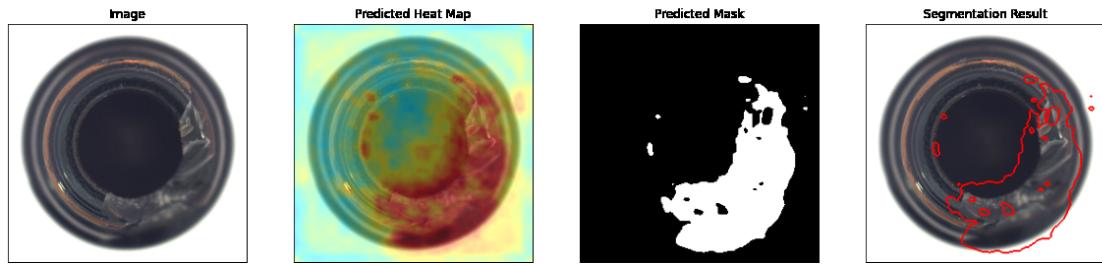


FIGURE 21 – Image anormale de la classe "bottle"

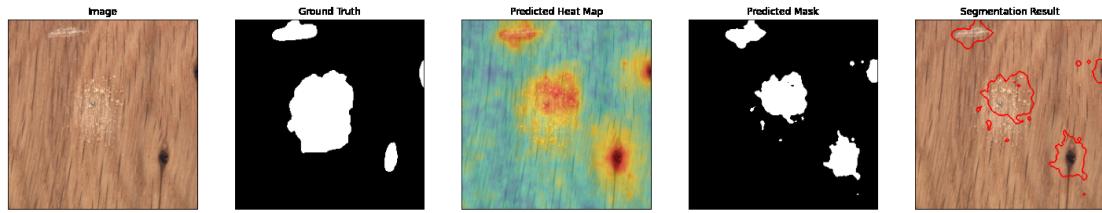


FIGURE 22 – Image anormale de la classe "wood" anomalie type rayure

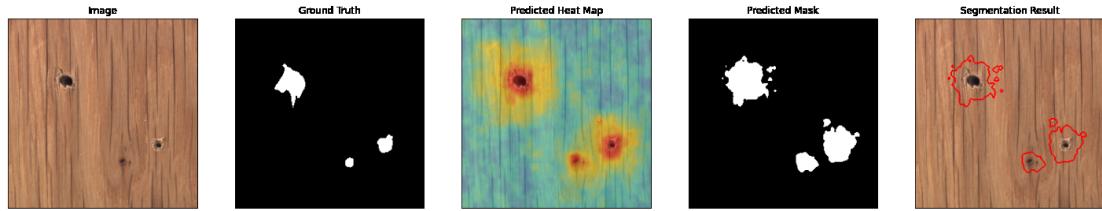


FIGURE 23 – Image anormale de la classe "wood"-anomalie type trou

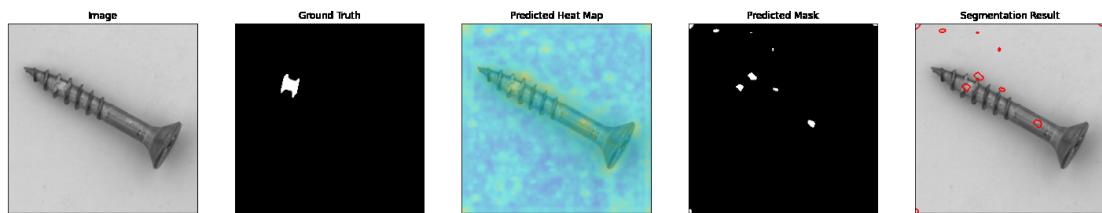


FIGURE 24 – Image anormale de la classe "screw"

De plus, on a testé le modèle FastFlow sur les classes pour lesquelles les modèles précédents n'étaient pas trop performants avec les hyperparamètres décrites dans l'état de l'art. La figure ci-dessous(cf Fig 25) montre la métrique de performance de modèle AUROC au niveau d'image en fonction de la classe d'objet.

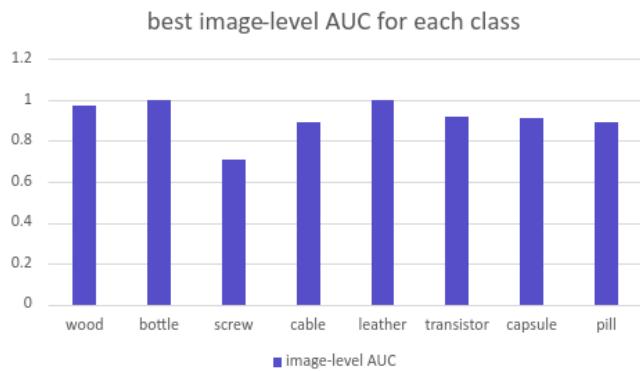


FIGURE 25 – Performance du modèle FastFlow sur plusieurs classes

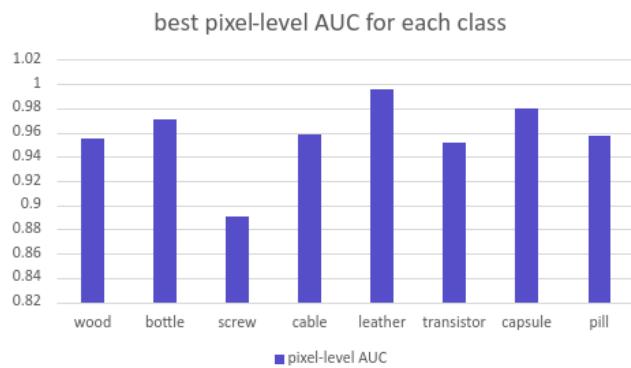


FIGURE 26 – Performance du modèle FastFlow sur plusieurs classes

Ces résultats sont très proches des valeurs indiquées dans l'état de l'art avec une mauvaise performance pour la classe "screw". La petite différence sera probablement causée par la technique early stopping utilisée comme méthode de régularisation lors de l'implémentation de code.

Dans le but de tester les hyperparamètres , on a choisi d'évaluer la performance du FastFlow en changeant la valeur de Batch Size et de Flow Step. Les deux figures ci-dessous (cf Fig 28) nous montrent les résultats obtenus.

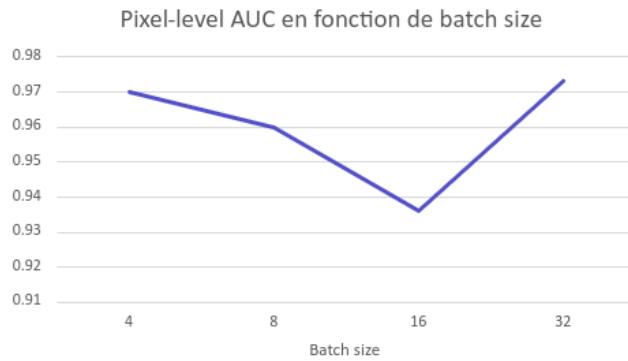


FIGURE 27 – Pixel-level AUC en fonction de batch size

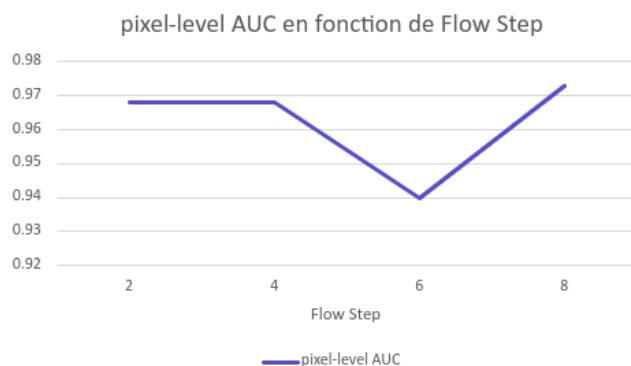


FIGURE 28 – Pixel-level AUC en fonction de flow step

On constate ici que les meilleures valeurs de batch size et Flow step sont respectivement 32 et 8 comme indiqué dans l'état de l'art en comparant la valeur de pixel-level Auc de notre modèle FastFlow pour la classe "bottle". Le nombre de step "8" sera expliqué par le fait que plus le nombre de flow step est élevé plus la distribution de probabilité sera proche de la distribution normale .Ainsi, la tâche de la détection d'anomalie sera simplifié.Faute d'espace mémoire, on n'a pas plus tester autres valeurs de flow step.

---

### 3 Conclusion

#### 3.1 Comparaison

##### 3.1.1 Résultats

Après les tests de chacun des modèles, voici les résultats obtenus :

Classes	Image ROCAUC				
	SPADE	PaDiM	PatchCore	RIAD	FastFLow
bottle	0.95	0.98	0.95	0.796	1.00
screw	0.667	0.74	0.995	0.983	0.706
wood	0.97	0.99	0.98	0.946	0.972

TABLE 7 – Les valeurs de ROCAUC de l'image pour les différents modèles

Classes	Pixel ROCAUC				
	SPADE	PaDiM	PatchCore	RIAD	FastFLow
bottle	0.644	0.97	0.93	0.854	0.971
screw	0.583	0.963	0.993	0.564	0.891
wood	0.562	0.93	0.97	0.985	0.955

TABLE 8 – Les valeurs de ROCAUC de pixel pour les différents modèles

D'après les résultats présentés dans les tableaux 7 8, PatchCore est le plus performant pour toutes les classes en termes de ROCAUC pour les images et pixels. FastFLow et PaDiM obtiennent également de bons résultats pour les classes bottle et wood, mais leurs performances sont moins bonnes pour la classe screw. Le modèle SPADE est moins performant pour la classe screw, tandis que RIAD est moins performant pour la classe bottle.

En étudiant les deux modèles PatchCore et PaDiM sur les données bruitées, nous pouvons confirmer que PatchCore a donné les meilleurs résultats pour toutes les classes sélectionnées grâce à des valeurs ROCAUC supérieures à 0,92, tandis que PaDiM présente de moins bonnes performances pour la classe screw.

##### 3.1.2 Analyse

Les résultats montrent que les performances des différents modèles varient en fonction de la classe et de la mesure de performance utilisée, il est donc important de conclure sur les forces et les limites de chaque modèle.

###### Forces et faiblesses de SPADE

- Le modèle SPADE ne nécessite aucun entraînement, ce qui le rend simple et rapide. Cependant, il semble être le moins performant parmi les algorithmes sélectionnés pour la segmentation d'anomalies. L'étude a été limitée à une carte de caractéristique d'un unique plus proche voisin, mais l'exploration de l'utilisation d'un plus grand nombre de voisins et d'un échantillonage plus important pourrait améliorer la performance de l'algorithme.

### Forces et faiblesses de PatchCore

- Grande précision : Le modèle PatchCore s'est avéré très performant sur MVTec.
  - Efficace : Le modèle est relativement efficace en termes d'exigences de calcul, comparé aux autres modèles de pointe. Il est donc plus facile de l'entraîner et de l'utiliser sur du matériel bas de gamme ou dans des situations où le calcul est limité.
  - Robuste : Le modèle s'est avéré résistant (faible baisse de performance) à divers types de corruptions et de perturbations de l'image, ce qui peut être important dans des situations réelles où les images ne sont pas toujours parfaitement propres.
- 
- Interprétabilité limitée : Comme de nombreux modèles d'apprentissage profond, le modèle PatchCore peut être difficile à interpréter et à comprendre, ce qui rend difficile le diagnostic des problèmes ou la compréhension de la façon dont il fait des prédictions.
  - Complexité de l'entraînement : Bien que le modèle puisse être efficace en termes d'exigences de calcul, il peut tout de même être difficile à entraîner,
  - Dépendances des ensembles de données : Les performances du modèle dépendent fortement de l'ensemble de données d'entraînement, et peuvent ne pas bien se généraliser à de nouveaux ensembles de données ou scénarios.

### Forces et faiblesses de FastFlow

- Malgré l'utilisation de la technique normalizing flow ces dernières années dans la génération d'images, le modèle FastFlow nous a fourni des bons résultats par rapport à ce qui est indiqué dans l'état de l'art. Ces résultats sont même comparables au modèle le plus preformant Patchcore.
- Il s'agit d'un modèle qui se montre assez performant vis à vis la tâche de localisation et détection d'anomalie grâce à l'orientation vers la technique normalizing flow 2D utilisée.
- À l'encontre des résultats de l'état de l'art, FastFLow n'était pas un bon modèle traitant la classe "Screw". Il a localisé des anomalies qui n'étaient pas même labellisées. De plus, on a rencontré parfois de mauvais cas de détection d'anomalie surtout lorsque les images présentent des tâches dans l'arrière plan.

### Forces et faiblesses de PaDiM

- PaDiM utilise la covariance et la moyenne pour modéliser la distribution des données normales, ce qui constitue une méthode efficace pour détecter les anomalies qui s'écartent de cette distribution. D'après nos résultats, PaDiM a donné des résultats intéressants pour différentes classes.
- Bien que PaDiM ait obtenu d'excellents résultats, il reste incapable de détecter efficacement des anomalies dans des données d'entrée bruitées.

---

## 4 Retour d'expérience

En premier lieu, ce projet a été pour nous l'occasion de découvrir la segmentation d'images, et plus particulièrement la segmentation d'anomalies.

À travers l'évaluation des différents algorithmes et modèles sur un jeu de données de référence, nous avons pu mettre en pratique nos connaissances préalablement mais aussi conjointement acquises. Concrètement, ce projet a été une opportunité pour approfondir nos connaissances théoriques vis à vis des GAN, WGAN, normalizing Flow, auto-encodeurs et autres algorithmes spécifiques, dans un contexte de recherche scientifique, par l'exploration et l'assimilation de papiers de recherche.

Par le partage de notre compréhension des modèles et les réunions récurrentes avec nos encadrants, nous avons enrichi notre savoir et nos compétences en matière de vision par ordinateur, sur un sujet particulièrement utile et intéressant.

Se pencher sur un challenge particulier et étudier les méthodes ayant obtenu les meilleurs résultats était également quelque chose de nouveau pour nous. Cela nous a donné un bon aperçu des projets et événements pouvant être organisés dans la communauté de la recherche en vision par ordinateur.

Pour avancer dans ce projet et fournir un code propre et compréhensible, nous avons fait le choix d'utiliser un répertoire *GitHub*<sup>1</sup>. Nous avons dû pour cela nous former à l'utilisation de cet outil, qui nous a permis d'avancer sereinement et progressivement tous ensemble, en rassemblant nos fichiers de travail.

Un gros point bloquant de notre projet a été le manque de capacités de calcul. La mise à disposition de crédits *Google Cloud Platform* assez compliqués à utiliser et pouvant facilement faire l'objet de mauvaises dépenses n'a pas été une réelle solution à ce problème : cela l'a transformé en problème chronophage, passant d'un soucis de ressources en calcul à un soucis de ressources en temps pour comprendre comment faire fonctionner ces nouvelles ressources en calcul.

En dernier lieu, ce travail en équipe a été un bon exercice pour que nous, jeunes étudiants ingénieurs, apprenions à travailler en équipe autonome pour répondre à un objectif désigné.

---

1. [github.com/ValColl/anomalydetection](https://github.com/ValColl/anomalydetection)

## Références

- [1] IVAN KOBYZEV, S. J. P., AND BRUBAKER, M. A. Normalizing flows : An introduction and review of current methods. *arXiv* :1908.09257v4 (2020).
- [2] JIAWEI YU, YE ZHENG, X. W. W. L. Y. W. R. Z. L. W. Fastflow : Unsupervised anomaly detection and localization via 2d normalizing flows. *arXiv* :2111.07677v2 (2021).
- [3] NIV COHEN, Y. H. Sub-image anomaly detection with deep pyramid correspondences,. *arXiv* :2005.02357 (2020).
- [4] ROTH, K., PEMULA, L., ZEPEDA, J., SCHÖLKOPF, B., BROX, T., AND GEHLER, P. Towards total recall in industrial anomaly detection. arxiv 2021. *arXiv preprint arXiv* :2106.08265.
- [5] T. DEFARD, A. SETKOV, A. L., AND AUDIGIER, R. Padim : a patch distribution modeling framework for anomaly detection and localization. *arXiv* :2011.08785 (2020).
- [6] ZAVRTANIK, V., KRISTAN, M., AND SKOČAJ, D. Reconstruction by inpainting for visual anomaly detection. *Pattern Recognition* 112 (2021), 107706.

---

## 5 Annexe

### 5.1 Annexe 1

#### Les résultats de référence du modèle PaDiM

Classe	Image ROCAUC	
	PaDiM ResNet18	PaDiM WideResnet50
bottle	0.981	0.983
zipper	0.982	0.985
wood	0.936	0.949
screw	0.974	0.985
carpet	0.98.9	0.991
tile	0.912	0.941

TABLE 9 – Les résultats de référence du modèle PaDiM.

Les couches	ROCAUC image sur toutes les classes
Couche 1	0.948
Couche 2	0.957
Couche 3	0.957
Couche 1+2+3	0.971

TABLE 10 – Les performances de référence du PaDiM en variant les couches d'extraction.

## 5.2 Annexe 2 : notes techniques

### 5.2.1 SPADE

L'algorithme SPADE[3] est un algorithme de segmentation d'anomalies basé sur une méthode k-plus proches voisins. Cet algorithme se décompose en 3 étapes :

1. extraction de caractéristiques ;
2. récupération d'une image normale par k-plus proches-voisins ;
3. alignement par pixel avec correspondances pyramidales profondes.

L'intuition générale est de reconstruire une image sans anomalie puis de comparer celle-ci avec l'image inférée pour en déduire les segments d'anomalies.

#### K-NN

Pour cette étape, un extracteur de caractéristiques de type ResNet pré-entraîné sur ImageNet est généralement utilisé.

#### Extraction des caractéristiques

Cette seconde étape est plus délicate. On commence par identifier les images contenant des anomalies. Pour cela, pour chaque image testée, la distance aux K images « normales » les plus proches est calculée. Si cette distance, calculée comme la distance euclidienne entre les caractéristiques extraites des images concernées, est supérieure à un seuil donné, alors cette image contient une anomalie.

#### Alignement d'images

Pour détecter les anomalies intra-images, l'objectif est de localiser les pixels d'une ou plusieurs anomalies contenues dans celle-ci. Afin d'éviter une forte dépendance aux variations parfois complexes des objets présents dans les images, on utilise une correspondance multi-images. Chaque pixel est comparé à l'aide des caractéristiques extraites à partir de celui-ci, et avec les K-plus proches images normales. Le pixel est détecté comme anormal si la distance est trop importante.

#### Alignement des pyramides de caractéristiques

Pour prendre en compte différents niveaux d'alignement, les caractéristiques issues des M derniers blocs de la pyramide ResNet sont utilisées et "alignées", c'est-à-dire que l'on compare les pixels correspondant aux mêmes endroits sur les images entre eux. Cela évite d'avoir à aligner directement les images, et permet d'avoir différents niveaux de contexte entrant en compte dans l'analyse.

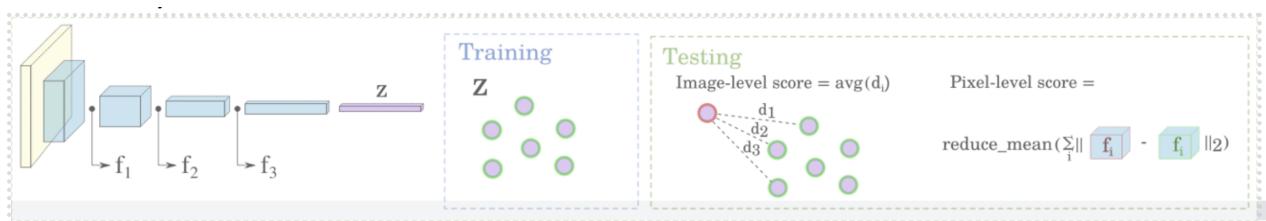


FIGURE 29 – Architecture

#### Résultats et avantages

Cette méthode atteint les performances « state-of-the-art » sur les jeux de données MVTec et Shaighai Tech Campus. Son gros avantage est qu'elle ne nécessite aucun entraînement : l'utilisation d'un réseau pré-entraîné pour extraire les caractéristiques suffit. C'est aussi une méthode extrêmement simple à mettre en place.

### 5.2.2 Description

#### L'entraînement :

Pour entraîner PaDiM, le modèle ResNet18 ou WideResNet50 est appliqué à N images d'apprentissage de produit "normal" afin de calculer la matrice et le vecteur des caractéristiques.

L'image suivante (cf Fig 30) illustre le processus d'apprentissage du modèle PaDim : nous passons au réseau neuronal CNN pré-entraîné avec une résolution (W,H) un ensemble de N images "normales" de produits, puis pour chaque patch d'image normale, nous associons un vecteur d'encodage de patch. Les vecteurs d'encodage de différents niveaux de couche sont ensuite concaténés pour obtenir des vecteurs d'encodage de patch portant des informations de différents niveaux sémantiques et résolutions. Ces vecteurs d'encodage sont utilisés pour estimer les matrices de covariance  $\Sigma$  et de moyenne  $\mu$  pour chaque position de la grille de patchs. Enfin, le résultat de l'apprentissage sera la matrice de covariance et la moyenne [5].

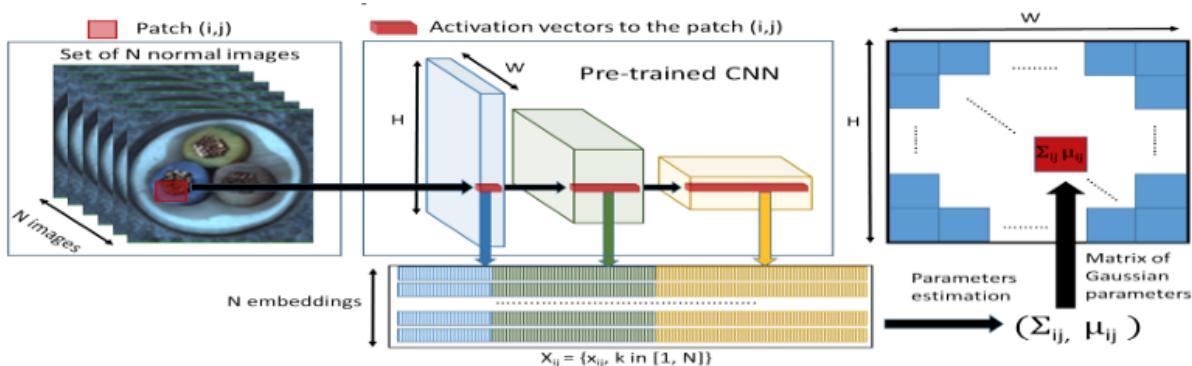


FIGURE 30 – Architecture du modèle PaDiM

#### L'inférence :

Pendant l'inférence du PaDiM, les vecteurs de caractéristiques sont calculés pour l'image d'entrée comme pendant l'entraînement. Ensuite, nous calculons la distance de Mahalanobis pour chaque pixel à partir de la matrice de covariance et de la moyenne précalculées. On utilise la distance de Mahalanobis  $\mathcal{M}(x_{ij})$  pour donner un score d'anomalie au patch en position  $(i, j)$  d'une image de test.  $\mathcal{M}(x_{ij})$  peut être interprétée comme la distance entre patch embedding de test  $x_{ij}$  et la distribution normale formée par les résultats de l'apprentissage  $\mathcal{N}(\mu_{ij}, \Sigma_{ij})$ , comme le montre l'équation suivante.

$$\mathcal{M}(x_{ij}) = \sqrt{(x_{ij} - \mu_{ij})^T \Sigma_{ij}^{-1} (x_{ij} - \mu_{ij})}$$

On peut donc calculer la matrice des distances de Mahalanobis  $\mathcal{M} = (\mathcal{M}(x_{ij}))_{1 < i < W, 1 < j < H}$  qui forme une carte des anomalies. Les scores élevés de cette carte indiquent les zones d'anomalie. Le niveau d'anomalie final de l'image entière est le maximum de la carte d'anomalie  $\mathcal{M}$ .

En utilisant les ensembles de données MVTec AD et STC, PaDiM a surpassé les méthodes existantes de détection et de localisation d'anomalies avec une valeur AUROC supérieure à 0,9.

### 5.2.3 Description

PatchCore est une technique de détection d'anomalies sur images qui se base sur un réseau neuronal pré-entraîné pour extraire des caractéristiques à partir de patches de l'image. Ces caractéristiques sont stockées dans une banque de mémoire de niveau patch pour tenir compte du voisinage. Cela se fait en plusieurs étapes (Fig 31).

Patchcore se base sur un modèle WideResNet50 pré-entraîné sur ImageNet. L'entraînement se fait uniquement sur des images labélisées (données nominales) pour détecter et segmenter les anomalies sur les images.

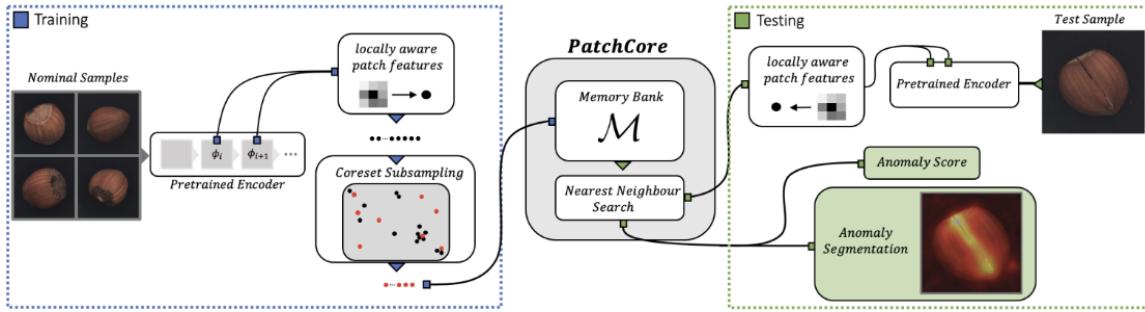


FIGURE 31 – Architecture de PatchCore

Les images d'entraînement passent par un réseau préentraîné sur ImageNet  $\phi$  (Encoder) pour avoir un point de départ. Cependant avec le préentraînement sur Imagenet, il peut y avoir une perte d'information sur les données nominales d'entrées et une introduction de biais.

#### Caractéristiques de patch sensibles au voisinage

A chaque image  $x_i$ , sont associées des caractéristiques  $\phi_{ij} = \phi_j(x_i)$  avec  $j$  le niveau d'hiérarchie dans le réseau pré-entraîné.  $\phi_{ij}$  est tenseur de  $h,w$  de profondeur  $c$ . Ces caractéristiques représentent un patch. Les patchs sont regroupés dans des collections de taille donnée  $p$  selon le nombre de patchs plus proches voisins défini et l'union de ces différentes collections forment la banque de données  $M$ . On a ainsi une mémoire de données formée de plusieurs collections de patchs dont le voisinage entre patchs est prise en compte (Fig 32).

---

#### **Algorithm 1:** PatchCore memory bank.

---

**Input:** Pretrained  $\phi$ , hierarchies  $j$ , nominal data  $\mathcal{X}_N$ , stride  $s$ , patchsize  $p$ , coresset target  $l$ , random linear projection  $\psi$ .

**Output:** Patch-level Memory bank  $M$ .

**Algorithm:**

```

 $M \leftarrow \{\}$ 
for  $x_i \in \mathcal{X}_N$  do
|    $M \leftarrow M \cup \mathcal{P}_{s,p}(\phi_j(x_i))$ 
end
/* Apply greedy coresset selection. */
 $M_C \leftarrow \{\}$ 
for  $i \in [0, \dots, l - 1]$  do
|    $m_i \leftarrow \arg \max_{m \in M - M_C} \min_{n \in M_C} \|\psi(m) - \psi(n)\|_2$ 
|    $M_C \leftarrow M_C \cup \{m_i\}$ 
end
 $M \leftarrow M_C$ 

```

---

FIGURE 32 – Algorithme de création de la banque de donnée  $M$  [<https://arxiv.org/pdf/2106.08265.pdf>]

#### CoreSet Subsampling

La banque de donnée M est sous échantillonnée avec la méthode de CoreSet Subsampling (sous échantillonage en gardant les mêmes propriétés) dans le but de reduire le temps de déduction tout en préservant les performances. Le but est de trouver des sous ensembles qui permettront d'approximer plus facilement la solution finale.

Pour rechercher les anomalies sur une image donnée  $x_{test}$ , le score d'anomalie est calculé en comparant la zone de test avec chaque voisin le plus proche dans la banque de mémoire et en déterminant la distance maximale. Si la distance maximale dépasse un seuil prédéfini, alors la zone de test est considérée comme anormale. La mesure d'évaluation ici est la courbe ROC (AUROC) qui montre à quel niveau le modèle est capable de classer les images de test.

Patchcore fait plus de 99% sur le dataset MVTEC AD selon la mesure AUROC. PatchCore recherche l'équilibre entre la préservation d'un maximum de contexte issu du réseau pré-entraîné sauvegardé dans la banque de donnée et le sous échantillonnage CoreSet pour augmenter la vitesse d'inférence lors du test.

Le modèle RIAD - Reconstruction by Inpainting for visual Anomaly Detection - se base sur la technique des mask autoencoders (cf figure 33). Il est très intéressant lorsqu'il faut identifier une zone qui fait dévier l'apparence habituelle d'un objet. Son fonctionnement suit les étapes suivantes :

1. découper chaque image en grille rectangulaires de taille  $k \times k$ ;
2. répartir aléatoirement en  $n$  sous échantillons chaque carreau de grille;
3. cacher ou non chaque sous échantillon avec une probabilité donné;
4. reconstruire l'image original à l'aide de l'autoencoder;
5. comparer l'image reconstruite et l'image originale pour identifier de potentielles zones de défauts.

L'autoencoder s'entraîne à reconstruire des images parfaites. S'il y a une différence entre l'image fournie en entrée et l'image reconstruire, il est probable que cette différence soit dû à une anomalie de l'image originale. Il est alors très facile d'identifier cette zone de défaut en faisant la différence des deux images. L'autoencoder peut être différent mais c'est un Unet qui a été choisi ici.

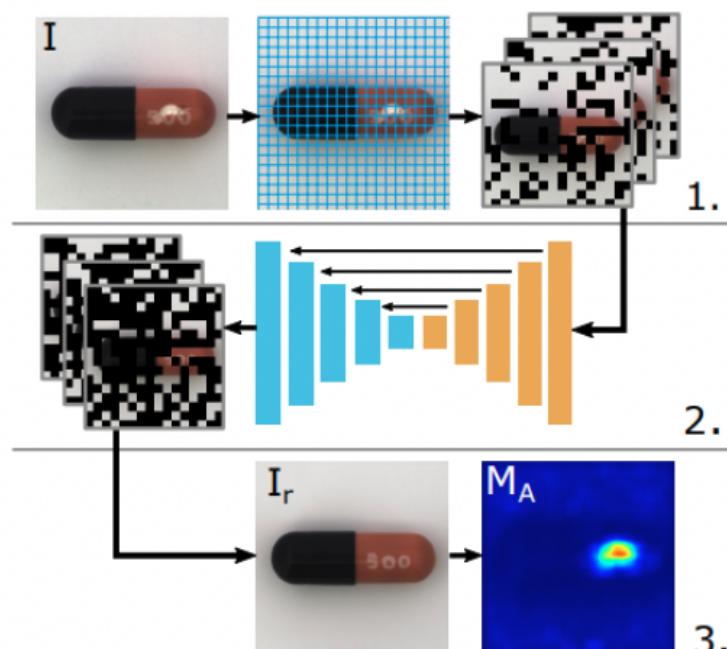


FIGURE 33 – Schéma de fonctionnement du modèle

Ce modèle peut être très intéressant dans certains cas car il souffre moins de sur apprentissage surtout dans le cas du challenge MVTEC. En effet, toutes les images originales sont très similaires et l'objectif du réseau est de reconstruire ces images parfaitement, ce qui constitue une tâche plus simple que de trouver une zone de défaut qui peut être située n'importe où.

### Hyperparamètres

Il y a plusieurs hyperparamètres sur lesquels jouer pour étudier ce modèle.

- la taille à laquelle on normalise toutes les images en entrée ;
- la taille de la grille  $k$ , en réalité plusieurs tailles sont prises et la moyenne des résultats est étudiée.
- le nombre de couches de l'autoencoder et leur nombre de filtre (cf fig.34);
- pourcentage de l'image caché (cf fig. 34);

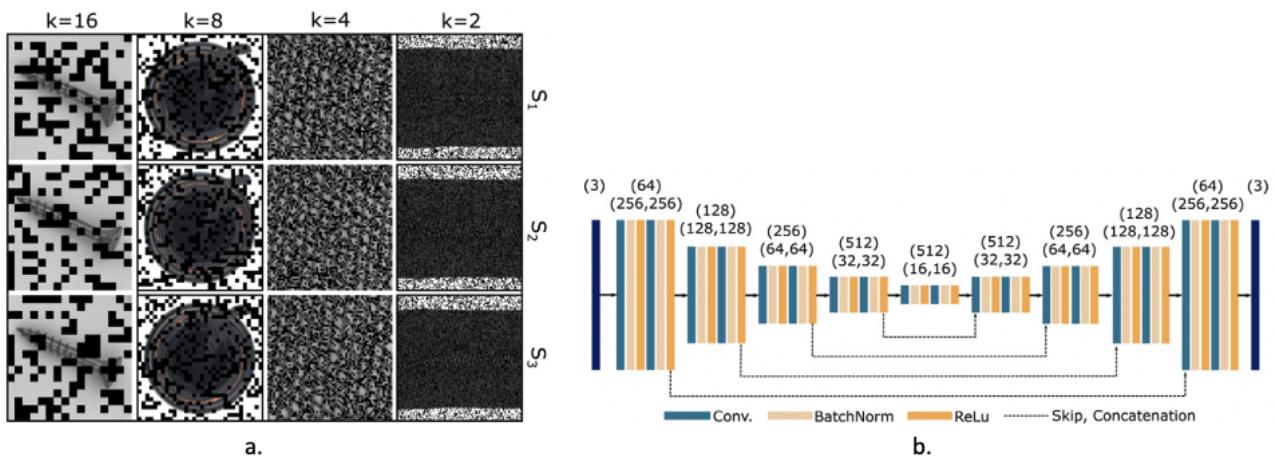


FIGURE 34 – Extrait des paramètres du modèle RIAD utilisé, venant de l'article [6] a : visualisation des différentes tailles de grilles utilisées, dont la moyenne des résultats est ensuite faite. b : Schéma de l'autoencodeur utilisé.

### 5.2.4 Description de modèle

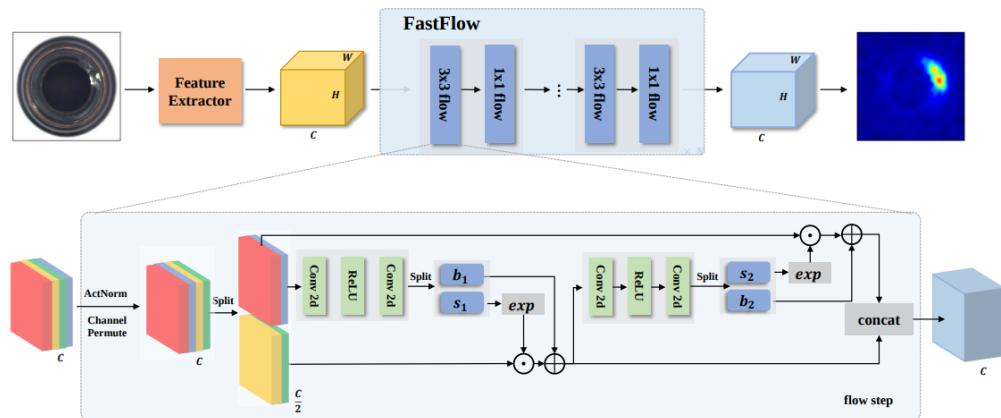


FIGURE 35 – Schéma représentatif de la structure de modèle

Extraction des caractéristiques L'étape indispensable pour commencer la tâche de détection d'anomalie est l'extraction des caractéristiques des images d'entrée normales. Il s'agit alors de l'utilisation de réseaux de neurones se basant sur les couches de convolution comme ResNet 18 pré-entraîné sur ImageNet qui a prouvé son efficacité vis à vis les résultats de test par rapport aux autres moyens d'extraction.

#### Estimation de la distribution d'entrée

Cet algorithme fait partie des "representation-based" méthodes puisqu'il estime la distribution des caractéristiques des images normales. À l'encontre des modèles d'estimation de distribution, Fastflow utilise la technique Normalizing Flow en 2D pour bien modéliser les caractéristiques globales et locales de l'image d'entrée 2D. Il s'agit alors de la projection des caractéristiques en haute dimension des images d'entrée normales obtenues par le réseau de neurones en une distribution normale (espace latent).

Détection et localisation des anomalies L'entraînement de ce modèle se fait seulement avec des images normales dont la distribution de probabilité de leurs caractéristiques extraites seront estimées par NF plus précisément par le block FastFlow 2D. Au moment de test, le modèle distingue les images d'anomalie de celles normales par un score qui se calcule par la distance entre les caractéristiques de l'image de test et la distribution estimée des caractéristiques normales. Plus précisément, les caractéristiques des images normales sont situées au centre de la distribution alors que celles des images anormales se trouvent à distance de ce centre. La performance de ce modèle se fait via la métrique "Area under the receiver operating characteristic curve (AUROC)" au niveau d'image et de pixel. Pour la tâche de détection d'anomalie, le modèle nous fournit un score d'anomalie pour chaque image de test alors que pour la tâche de localisation c'est le score d'anomalie pour chaque pixel qui est donné.

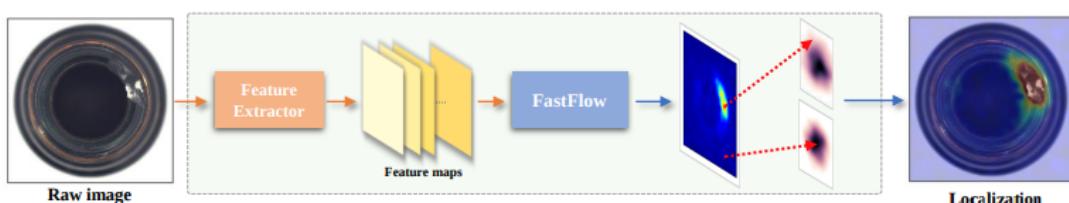


FIGURE 36 – Schéma représentatif de détection d'anomalie lors de test