

Résumé

Dans ce mémoire, nous proposons une méthodologie qui permet de détecter des anomalies parmi un ensemble de données complexes, plus particulièrement des images. Pour y arriver, nous utilisons un type spécifique de réseau de neurones, soit un autoencodeur variationnel (VAE). Cette approche non-supervisée d'apprentissage profond nous permet d'obtenir une représentation plus simple de nos données sur laquelle nous appliquerons une mesure de distance de Kullback-Leibler nous permettant de discriminer les anomalies des observations "normales". Pour déterminer si une image nous apparaît comme "anormale", notre approche se base sur une proportion d'observations à filtrer, ce qui est plus simple et intuitif à établir qu'un seuil sur la valeur même de la distance. En utilisant notre méthodologie sur des images réelles, nous avons démontré que nous pouvons obtenir des performances de détection d'anomalies supérieures en termes d'aire sous la courbe (ROC), de précision et de rappel par rapport à d'autres approches non-supervisées. De plus, nous avons montré que la simplicité de l'approche par niveau de filtration permet d'adapter facilement la méthode à des jeux de données ayant différents niveaux de contamination d'anomalies.

Abstract

In this master's thesis, we propose a methodology that aims to detect anomalies among complex data, such as images. In order to do that, we use a specific type of neural network called the variational autoencoder (VAE). This non-supervised deep learning approach allows us to obtain a simple representation of our data on which we then use the Kullback-Leibler distance to discriminate between anomalies and "normal" observations. To determine if an image is considered as "abnormal", our approach is based on a proportion of observations to be filtered, which is easier and more intuitive to establish compared to applying a threshold based on the value of a distance metric. By using our methodology on real complex images, we can obtain superior anomaly detection performances in terms of area under the curve (AUC), precision and recall compared to other non-supervised methods. Moreover, we demonstrate that the simplicity of our filtration level allows us to easily adapt the method to datasets having different levels of anomaly contamination.

Table des matières

Résumé	i
Abstract	ii
Table des matières	iii
Liste des tableaux	iv
Liste des figures	v
Remerciements	ix
Introduction	1
1 Contexte	3
1.1 Les méthodes de détection d'anomalies	3
1.2 Les autoencodeurs	8
1.3 Les autoencodeurs variationnels	12
2 Méthodologie	17
2.1 Les objectifs de l'approche	17
2.2 Les hypothèses de l'approche	18
2.3 Description de l'approche	19
3 Expérimentations	28
3.1 Jeux de données	28
3.2 Méthodes testées	30
3.3 Résultats	36
3.4 Discussion	42
Conclusion	55
A <Titre de l'annexe>	57
Bibliographie	58

Liste des tableaux

3.1	Description des 2 ensembles de données pour le jeu de données de <i>ImageNet</i>	29
3.2	Description des 2 ensembles de données pour le jeu de données provenant de <i>MNIST</i>	30
3.3	Description des 2 ensembles de données pour le jeu de données de <i>MNIST</i>	30
3.4	Horaire des paramètres β pour le jeu de données provenant de <i>ImageNet</i> et le jeu de données provenant de <i>MNIST</i>	33
3.5	Résultats selon les métriques d'aire sous la courbe ROC, de précision et de rappel pour les différentes expérimentations concernant le jeu de données <i>ImageNet</i>	37
3.6	Résultats des aires sous la courbe ROC des différentes approches selon le scénario de contamination et le scénario de test appliquée sur le jeu de données <i>MNIST</i>	39
3.7	Résultats des précisions selon la quantité de contamination et le scénario de test sur le jeu de données <i>MNIST</i> . La valeur de α dépend du scénario de contamination. Pour le scénario "Moins" $\alpha = 0.01$, pour le scénario "Égal" $\alpha = 0.05$ et finalement pour le scénario "Plus" $\alpha = 0.1$	40
3.8	Résultats des rappels selon la quantité de contamination et le scénario de test sur le jeu de données <i>MNIST</i> . La valeur de α dépend du scénario de contamination. Pour le scénario "Moins" $\alpha = 0.01$, pour le scénario "Égal" $\alpha = 0.05$ et finalement pour le scénario "Plus" $\alpha = 0.1$	41

Liste des figures

1.1	Distribution de 5000 simulations d'une loi normale et d'une loi standard Cauchy	4
1.2	Exemple illustrant la structure de base d'un autoencodeur. Dans le cas ci-dessus, on pourrait interpréter le schéma comme un réseau pleinement connecté où les blocs pourraient représentés les neurones et les liens seraient les poids, ou les paramètres du réseau. Le concept s'applique également à une architecture de réseau à convolutions, où les paramètres appris sont les filtres de convolutions.	9
1.3	Exemple simpliste d'une architecture d'autoencodeur	10
1.4	Structure de base d'un autoencodeur variationnel. La représentation latente z est générée en simulant de la loi $q_\theta(z x)$, qui suit une loi normale multivariée avec les paramètres μ et σ calculés par le réseau. C'est de là que vient la composante stochastique du modèle.	13
2.1	Figure illustrant notre méthodologie pour transformer une image d'entrée brute en format vectoriel selon le nombre de canaux de l'image. Les valeurs présentées dans les matrices sont fictives.	19
2.2	Figure illustrant des exemples d'images considérées comme "normales" et d'autres comme "anormales". Dans ce cas, les images dites "normales" sont des images de voitures.	21
2.3	Figure montrant le mécanisme derrière le concept de <i>perceptual optimization</i> . Au lieu de calculer la perte entre \mathbf{x} et $\hat{\mathbf{x}}$, la perte est calculée entre $g(\mathbf{x})$ et $g(\hat{\mathbf{x}})$. La fonction $g(a)$ permet d'extraire les valeurs d'une couche spécifique d'un autre réseau pré-entraîné, comme par exemple <i>VGG16</i> pré-entraîné sur <i>ImageNet</i>	22
2.4	Exemples de situations représentant les 2 différents scénarios se rapportant aux représentation latentes des observations "normales" et "anormales".	25
3.1	Exemples d'images pour notre jeu de données d'images réelles tirées de <i>ImageNet</i> . La figure à gauche (a) correspond à des exemples de la classe dite "normale" et le volet de droite (b) correspond à des exemples de la classe d'anomalies.	29
3.2	Exemples d'images pour notre jeu de données <i>MNIST</i> . La figure à gauche (a) correspond à des exemples de la classe "normale" et le volet de droite (b) correspond à des exemples de la classe d'anomalies. Il s'agit plus particulièrement du scénario de test 3.	30
3.3	Architecture du modèle DA-VAE pour le jeu de données <i>ImageNet</i> . Les blocs représentent les couches du réseau. Les blocs jaunes représentent les couches où il n'y a pas de paramètres à optimiser. Les blocs verts sont des couches de convolution ou déconvolution et les blocs gris sont des couches linéaires.	32

3.4	Architecture du modèle DA-VAE pour le jeu de données provenant de <i>MNIST</i> . Les blocs représentent les couches du réseau. Les blocs jaunes représentent les couches où il n'y a pas de paramètres à optimiser. Les blocs verts sont des couches de convolution ou déconvolution et les blocs gris sont des couches linéaires.	32
3.5	Graphiques en boîtes et moustaches illustrant les résultats sur les 20 expérimentations de chacune des approches et scénarios de contamination. Les 3 sous-figures concernent une métrique différente.	38
3.6	Graphiques en boîtes et moustaches illustrant les résultats sur les 20 expérimentations de chacune des approches, des scénarios de test et des scénarios de contamination. Toutes les figures présentent les résultats en aire sous la courbe ROC.	40
3.7	Graphiques en boîtes et moustaches illustrant les résultats sur les 20 expérimentations de chacune des approches, des scénarios de test et des scénarios de contamination. Toutes les figures présentent les résultats en précision.	41
3.8	Graphiques en boîtes et moustaches illustrant les résultats sur les 20 expérimentations de chacune des approches, des scénarios de test et des scénarios de contamination. Toutes les figures présentent les résultats en rappel.	42
3.9	Graphiques illustrant les scores d'anomalies selon le scénario de contamination pour le modèle DA-VAE appliqué sur le jeu de données <i>ImageNet</i> . Les points violet sont des observations que nous connaissons comme "normales" alors que les points jaunes sont des observations que nous connaissons comme "anormales". Dans tous les cas, le niveau de filtration α est défini comme le niveau de contamination dans le jeu de données de test.	44
3.10	Échantillons d'images provenant du jeu de données d' entraînement ayant des statistiques de distance faibles (a) et élevées (b) pour le scénario "Plus" du jeu de données <i>ImageNet</i>	45
3.11	Échantillons d'images pré-sélectionnées avec l'inverse de leur score d'anomalie $(1 - \gamma)$. Le score a été calculé selon le modèle DA-VAE du scénario "Plus" sur le jeu de données <i>ImageNet</i>	45
3.12	Moyenne des μ et σ des représentations latentes du jeu de données d' entraînement sur le jeu de données <i>ImageNet</i>	46
3.13	Graphique des 2 premières composantes principales réalisé sur les vecteurs μ et σ du jeu de données d' entraînement de <i>ImageNet</i> appliquée sur le scénario de contamination "Plus".	47
3.14	Pourcentage du critère de Kullback-Leibler dans la perte totale selon l'itération d' entraînement et le scénario de contamination. Ces résultats sont tirés du scénario de contamination "Plus" pour le jeu de données <i>ImageNet</i>	48
3.15	Exemples d'images générées par le modèle DA-VAE en utilisant seulement la partie décodeur de l'autoencodeur ainsi que des simulations provenant d'une loi $N(0, I)$	48
3.16	Graphiques illustrant les scores d'anomalies selon le scénario de contamination pour le modèle DA-VAE appliquée sur le scénario de test 1 de <i>MNIST</i> . Les points violet sont des observations que nous connaissons comme "normales" alors que les points jaunes sont des observations que nous connaissons comme "anormales". Dans tous les cas, le niveau de filtration α est défini comme le niveau de contamination dans le jeu de données de test.	50

3.17 Échantillons d'images provenant du jeu de données d'entraînement ayant des statistiques de distance faibles (a) et élevées (b) pour le scénario de test 3 ("Plus") du jeu de données <i>MNIST</i>	51
3.18 Moyenne des μ et σ des représentations latentes du jeu de données d'entraînement sur le jeu de données <i>MNIST</i>	52
3.19 Graphique des 2 premières composantes principales réalisé sur les vecteurs μ et σ du jeu de données d'entraînement <i>MNIST</i> appliqué sur le scénario de test 3 et le scénario de contamination "Plus".	53
3.20 Pourcentage du critère de Kullback-Leibler dans la perte totale selon l'itération d'entraînement et le scénario de contamination. Ces résultats sont tirés du scénario de test 3, où le chiffre "1" est considéré comme la classe "normale" et tous les autres sont considérés dans la classe "anormale".	54

<Dédicace si désiré>

Remerciements

Introduction

La détection d'anomalie est un sujet complexe qui a généré beaucoup de littérature en statistique, en apprentissage machine et plus récemment en vision numérique. Il existe plusieurs applications à ce sujet dans les domaines de la cyber-intrusion, de la finance et de l'assurance, de la médecine ou dans l'identification de dommages industriels (Chandola et al., 2007). Une anomalie est définie par Zimek and Schubert (2017) comme un événement, une mesure ou une observation qui diffère significativement de la majorité des données. Une anomalie, ou également appelée une aberration, est un concept intrinsèque à plusieurs domaines reliés à l'analyse de données, car une telle donnée est généralement intéressante à identifier ou retirer d'une source de données. Il peut être intéressant de l'identifier simplement parce que c'est la tâche poursuivie. Il peut également être intéressant de la retirer avant de réaliser une autre tâche d'apprentissage.

Une difficulté reliée à la détection d'anomalie est qu'on doit souvent utiliser des données non étiquetées, car les anomalies sont généralement générées par des phénomènes imprévus. Cela fait en sorte que le problème devient non-supervisé. Une autre difficulté, plus particulièrement reliée avec les approches non-supervisées, est qu'il faut généralement déterminer un seuil qui nous permet de prendre une décision quant à la nature d'une donnée (anomalie ou non). Finalement, ces difficultés deviennent encore plus prononcées lorsqu'on doit traiter des données complexes, comme des images.

Dans le contexte de données à hautes dimensions, les réseaux de neurones sont couramment utilisés. En effet, leurs couches superposées peuvent partir d'une entrée complexe, comme une image, et compresser cette information vers une représentation vectorielle plus simple et riche en informations. Les autoencodeurs sont une catégorie de réseaux de neurones fréquemment utilisés pour traiter un problème non-supervisé. Il existe d'ailleurs plusieurs applications d'autoencodeurs dans un contexte de détection d'anomalie, où l'erreur de reconstruction est souvent utilisée comme indicateur d'anomalie. Cependant, ces méthodes requièrent de trouver un seuil, souvent sous forme de distance ou de métrique, qui peut être difficile à établir ou à expliquer. C'est d'ailleurs pour cette raison que An and Cho (2015) proposent de se baser

sur une probabilité de reconstruction, qui est un mesure plus objective et ne requiert pas de seuil quelconque. Par contre, cette mesure de probabilité est basée sur la capacité de reconstruction, qui peut être problématique dans le contexte d'images complexes, plutôt que sur la représentation latente, qui elle est plus simple.

Dans cette étude, nous proposons une approche qui vise à simplifier la détermination du seuil nécessaire pour la détection d'anomalie à partir de données non étiquetées. Avec cette contribution, nous proposons d'utiliser des méthodes existantes comme les autoencodeurs et les tests d'hypothèses pour encoder des structures de données complexes dans un cadre décisionnel simple et intuitif. À partir d'autoencodeurs variationnels (VAE), nous sommes en mesure de créer ces dites représentations et de détecter les anomalies à partir d'un niveau de confiance, plutôt que d'une métrique ou une distance.

Chapitre 1

Contexte

En premier lieu, nous allons commencer par faire une brève revue des différentes catégories de méthodes de détection d'anomalies et de leur fonctionnement respectif. Ensuite, nous allons faire un résumé de la théorie des autoencodeurs et comment ceux-ci sont pertinents dans un contexte de détection d'anomalies. Ensuite, nous allons décrire plus en détails un type particulier d'autoencodeur utilisé dans cette étude, soit l'autoencodeur variationnel. Finalement, nous allons couvrir quelques notions de base quant aux tests d'hypothèses, un cadre statistique classique pour prendre des décisions.

1.1 Les méthodes de détection d'anomalies

Tout d'abord, il est pertinent de commencer par mentionner que toutes les méthodes de détection d'anomalies fonctionnent fondamentalement de la même manière. En effet, ces algorithmes sont en mesure de faire l'apprentissage d'un jeu de données et de stocker cette information dans un modèle d'apprentissage. En sachant ce qui est normal, il est donc également possible d'utiliser ce modèle pour évaluer ce qui est anormal en identifiant ce qui dévie de cette normalité. Le choix du modèle est cruciale, car si celui-ci ne s'ajuste pas bien aux données, il pourrait nous induire en erreur sur l'anormalité d'une observation (Aggarwal, 2016).

Pour démontrer l'importance du choix de modèle, prenons un exemple simpliste et fréquemment utilisé en pratique, soit sur la moyenne d'une loi normale à variance connue, souvent appelé le test Z . Dans ce test statistique, qui peut être utilisé pour de la détection d'anomalies, l'hypothèse nulle est que les données suivent une loi normale. Prenons par exemple des observations à 1 dimension X_1, \dots, X_N avec de moyenne μ et d'écart-type σ . La valeur Z pour une observation X_i est donnée par :

$$Z_i = \frac{|X_i - \mu|}{\sigma}. \quad (1.1)$$

Cette valeur Z nous donne en fait le nombre d'écart-types dont une observation dévie de la moyenne. On peut généralement penser que si une observation dévie beaucoup de la moyenne, ça peut être un indicateur important d'anomalie. Dans ce cas-ci, on utilise généralement la règle du pouce $Z_i \geq 3$ comme indicateur d'anomalie. En d'autres mots, une observation est improbable, ou potentiellement "anormale", si elle se situe à plus ou moins 3 écarts-types de la moyenne. Ce modèle de détection d'anomalie s'applique bien dans le cas où nos observations proviennent réellement d'une loi normale. Dans la figure 1.1a, on peut voir 5000 simulations d'une loi normale de moyenne nulle et d'écart-type égale à 2. On peut voir que notre règle du pouce, ou notre modèle, fonctionne relativement bien puisque les valeurs se situant à l'extérieur de l'intervalle $[-6, 6]$ ont tous une fréquence relative très faible. De la même manière, les valeurs se situant à l'intérieur de l'intervalle $[-6, 6]$ ont des fréquences relatives acceptables ($\geq 0.5\%$). Cela nous laisse croire que la détection d'anomalies est adéquate. Dans une autre situation où les données ne proviennent pas d'une loi normale, notre règle du pouce, ou notre modèle, peut être moins adéquat. Supposons que nos données proviennent plutôt d'une loi à queue lourde, comme la loi standard Cauchy. En simulant 5000 observations de cette loi, nous obtenons une moyenne empirique de -1.6 et un écart-type empirique de 160.2 . C'est donc dire que l'intervalle défini avec le modèle précédent serait de $[-482.2, 479]$. On peut voir à la figure 1.1b que plusieurs observations ne sont pas considérées comme des anomalies selon ce modèle, alors que leur fréquence relative est pourtant très faible (par exemple inférieure à 0.5%).

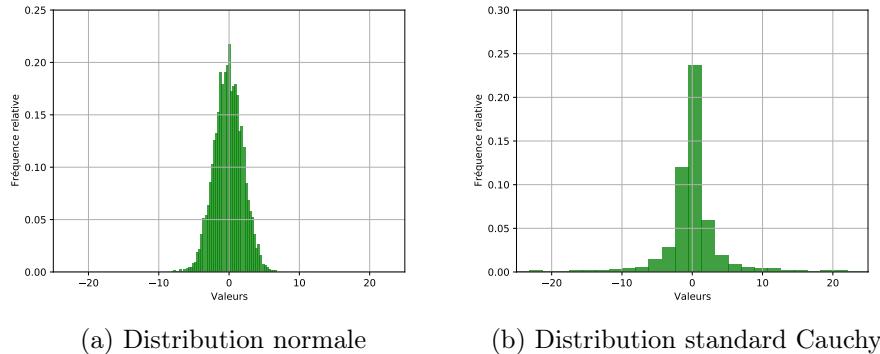


FIGURE 1.1 – Distribution de 5000 simulations d'une loi normale et d'une loi standard Cauchy

L'exemple du test Z illustre simplement le fait que le choix du modèle aura un impact important dans la détection d'anomalies. Au final, la pertinence du modèle choisi sera étroitement lié à la nature des données. Il existe plusieurs catégories d'algorithmes de détection d'anomalies. Dans Aggarwal (2016), 4 de ces approches nous apparaissent comme particulièrement intéressantes à survoler.

1.1.1 Analyse des valeurs extrêmes

La détection d'anomalies via l'analyse des valeurs extrêmes est probablement une des approches les plus simples. Dans un cas à une seule dimension, les anomalies sont définies comme les valeurs très grandes ou très faibles par rapport à la majorité des autres valeurs. On s'intéresse donc à la queue d'une distribution, comme dans l'exemple avec le test Z présenté plus tôt. Cette approche vient légèrement à l'encontre de la définition d'une anomalie présentée plus tôt où l'accent est plutôt mis sur le fait qu'une anomalie diffère de la majorité des données. Cette dernière définition fait davantage référence à une notion de probabilité.

Bien que l'analyse des valeurs extrêmes ne soit pas la méthode la plus utilisée en pratique pour identifier des anomalies, elle fait souvent partie intégrante de plusieurs autres méthodes. En effet, l'analyse de valeurs extrêmes est souvent utilisée comme étape finale de plusieurs autres algorithmes. C'est d'ailleurs ce qui permet généralement de déterminer ce qui dévie d'une certaine normalité, via une valeur unidimensionnelle comme un score d'anomalie ou une distance.

1.1.2 Les modèles probabilistes

L'élément clé des approches basées sur des modèles probabilistes est qu'on fait une hypothèse sur la distribution des données. Ensuite, on ajuste le modèle statistique correspondant à cette hypothèse sur les données en faisant l'apprentissage des paramètres du modèle. Un exemple classique pourrait être d'ajuster un mélange de k lois normales. Avec ce modèle, on fait l'hypothèse que chaque observation provient d'un des k groupes de la loi mélange. On peut faire l'apprentissage des paramètres du modèle avec l'algorithme *expectation-maximization (EM)*. Par la suite, on peut évaluer l'anormalité d'une observation en se basant sur la probabilité de cette instance selon notre modèle ajusté. Ce type d'approche a l'avantage de pouvoir s'appliquer assez facilement à plusieurs types de données. Par contre, le désavantage est que l'on doit faire une hypothèse quant à la distribution des données, qui peut parfois être inadéquate. Certains jeux de données peuvent difficilement être représenté par une distribution connue, ce qui peut mener à un mauvais ajustement du modèle et ainsi tirer de mauvaises conclusions quant à la nature d'une observation.

1.1.3 Les modèles linéaires et non-linéaires

Ce type d'approche est basé sur l'apprentissage d'un espace à plus petites dimensions via un modèle linéaire ou non-linéaire. Un exemple classique est l'utilisation d'une régression linéaire. Par exemple, dans un cas d'une régression linéaire à 2 variables explicatives, l'apprentissage d'une variable réponse y_i est donnée par l'équation 1.2. Dans ce cas-ci, on utilise généralement la méthode des moindres carrés pour trouver les paramètres optimaux du modèle, soit les paramètres β_0 , β_1 et β_2 .

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \epsilon_i \quad \forall i \in 1, \dots, N \quad (1.2)$$

Les estimations de ces paramètres, soit $\hat{\beta}_0$, $\hat{\beta}_1$ et $\hat{\beta}_2$, nous permettent d'estimer la variable réponse \hat{y}_i :

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1,i} + \hat{\beta}_2 x_{2,i} \quad \forall i \in 1, \dots, N \quad (1.3)$$

On peut ensuite utiliser l'erreur de prédiction comme indicateur d'anomalie. Cette erreur de prédiction, défini comme $\hat{y}_i - y_i$, est également appelée erreur de reconstruction. La prémissse de base est que si le modèle est en mesure de bien ajuster les données, les observations normales devraient être bien reconstruites par le modèle. À l'inverse, les observations anormales devraient être mal reconstruites par le modèle. On peut d'ailleurs ce servir encore une fois de l'analyse des cas extrêmes pour reconnaître les erreurs les plus importantes, et par le fait même, les anomalies.

Un autre exemple classique dans cette catégorie d'approches est l'utilisation d'une analyse en composante principale (ACP). Dans ce cas-ci, nous n'avons pas de variable réponse y_i sur laquelle calculer une erreur de reconstruction. L'erreur de reconstruction sera donc basée sur la capacité de reconstruire l'entrée x_i . Cette technique de réduction de dimensionnalité permet de transformer les données vers un espace où les variables sont décorrélées et où l'utilisation d'un sous-ensemble de ces variables latentes peut être suffisant pour expliquer une bonne partie de la variabilité des données originales. La transformation permettant d'obtenir un espace latent à k dimensions est donnée par l'équation 1.4.

$$Z = X V \quad (1.4)$$

Dans l'équation 1.4, X est une matrice $n \times p$ représentant les données initiales et où le vecteur de moyennes $\boldsymbol{\mu}$, calculé sur chaque colonne, a été soustrait à la matrice originale. X est donc la matrice centrée des données initiales. V représente une matrice $p \times k$, où $k < p$, de vecteurs propres correspondant aux valeurs propres les plus élevées de la matrice de corrélation Σ . La valeur k signifie qu'on conserve seulement k variables latentes, ou également appelées composantes principales. La reconstruction de cet espace latent à k dimensions peut ensuite être retrouvé par l'équation 1.5.

$$\hat{X} = Z V^\top + \boldsymbol{\mu} \quad (1.5)$$

La projection de cet espace latent vers l'espace original défini par l'équation 1.5 permet d'obtenir une erreur de reconstruction. Celle-ci est peut être obtenue par une fonction appliquée entre \hat{X} et X . Par exemple, on pourrait prendre la distance quadratique entre chaque instances de \hat{X} et X et ainsi obtenir n erreurs de reconstruction. Cette erreur de reconstruction peut être utilisée pour trouver les observations qui dévie de la normalité du modèle de la même manière qu'avec une régression linéaire.

Cette catégorie d'approches de détection d'anomalies a le potentiel de mieux s'adapter à des données où il n'y a pas de distribution connue. La régression linéaire est un exemple de modèle linéaire, mais on pourrait également utiliser un modèle plus complexe qui permet de capture des non-linéarités. Par exemple, il est possible d'utiliser des réseaux de neurones pour faire l'apprentissage des données vers un espace à plus basse dimension. Le désavantage avec de telles approches est qu'on peut perdre la notion d'interprétabilité. Dans certains exemples d'applications, il peut être intéressant de savoir quelles raisons expliquent la présence d'une anomalie dans les données.

1.1.4 Les méthodes basées sur les distances

Cette catégorie de méthodes de détection d'anomalie est basée sur l'idée de trouver les observations qui sont isolées de la majorité des autres observations. Cela est généralement quantifié par des mesures de distances ou de dissimilarités. Parmi ces méthodes, on peut retrouver 3 sous-catégories fréquemment utilisées en pratique : les regroupements (*clustering*), les méthodes de densité et les méthodes des plus proches voisins.

Dans le cas des regroupements et des méthodes de densité, l'objectif est de trouver des zones de l'espace qui caractérisent le jeu de données. Les anomalies sont généralement identifiées en considérant les observations qui ne font pas parties de ces zones. Dans le cas spécifique des regroupements, on fait un séparation de l'espace qui est basée sur les observations elles-mêmes. Si on prend comme exemple l'algorithme de regroupement k -moyennes, on peut évaluer le score d'anomalie d'une observation en prenant la distance minimale d'un centroïde trouvé pendant l'ajustement du modèle.

Dans les méthodes de densité, plutôt que séparer l'espace par l'entremise d'observations, on sépare directement des zones de cette espace. Ensuite, on évalue la densité des observations dans une zone selon le nombre d'observations dans cette même zone. Le fait de séparer l'espace des données nous permet de mieux quantifier la densité d'un point, peu importe son emplacement dans l'espace. Un exemple simple serait d'utiliser la méthode de l'histogramme afin de compartimenter l'espace en plusieurs sous-espaces et ensuite évaluer la densité d'une région par le nombre d'observations s'y retrouvant. Ce genre d'approche a généralement l'avantage d'être interprétable, car on peut savoir exactement pourquoi une observation fait partie d'un sous-ensemble ayant une faible densité. Prenons un exemple simple à seulement 2 dimensions

où on retrouve le poids et la grandeur d'une personne. En séparant l'espace en différentes régions, on réalise que seulement très peu de personnes ont un poids inférieur à 45 kg et une grandeur supérieur à 1.80 m. Ainsi, on peut facilement expliquer pourquoi une personne de 39 kg et de 1.85 m est considérée comme une anomalie.

Finalement, les méthodes basées sur les plus proches voisins permettent de définir un score d'anomalie qui dépend de la distance entre une observation et ses k plus proches voisins. Plus cette distance est grande, plus on peut penser que la donnée est isolée du reste des autres observations. Encore une fois, on peut avoir recours à l'analyse des cas extrêmes pour évaluer à partir de quelle distance on peut considérer une donnée comme une anomalie.

1.2 Les autoencodeurs

Maintenant que nous avons couvert certaines notions de base par rapport à différentes approches de détection d'anomalie, nous allons couvrir la théorie de base derrière le fonctionnement des autoencodeurs. Comme mentionné dans la section 1.1.3, certaines approches de modélisation plus complexes, comme les autoencodeurs, peuvent être utilisées pour faire l'apprentissage d'un jeu de données.

Les premiers travaux se rapprochant de l'autoencodeur connu aujourd'hui remontent aux années 1980 (Rumelhart et al., 1986). Dans ses travaux, le principe était d'apprendre une représentation cachée en utilisant l'entrée comme variable réponse. Avec la montée en popularité des réseaux de neurones au début des années 2000, une nouvelle génération d'autoencodeurs ont vu le jour. Ces autoencodeurs avaient désormais plusieurs couches cachées superposées (Hinton and Salakhutdinov, 2006), permettant ainsi de faire l'apprentissage de données plus complexes comme des images. Un autoencodeur est un réseau de neurones qui a comme objectif d'apprendre une représentation intermédiaire et efficiente d'une entrée de manière non-supervisée (Goodfellow et al., 2016). Pour réaliser cette objectif, l'autoencodeur se décompose en 2 composantes : un encodeur et un décodeur. L'encodeur reçoit en entrée x et convertit celui-ci vers une représentation latente z . Le décodeur prend en entrée cette représentation latente z et la décide pour ainsi retrouver le plus possible l'entrée initiale x . Cette structure de base est illustrée dans la figure 1.2. Historiquement, les autoencodeurs étaient vus comme une méthode de réduction de dimensionnalité, mais désormais ceux-ci ont davantage d'applications dû au fait qu'ils peuvent apprendre des variables latentes riches en informations. On peut citer des applications en vision numérique pour retrouver le contenu d'une image (Krizhevsky and Hinton, 2011) ou même en traitement de la langue naturelle pour interpréter un discours oral (Feng et al., 2014).

L'intuition derrière les autoencodeurs est essentiellement de reconstruire une entrée x en passant par 2 fonctions (l'encodeur et le décodeur) apprises par le modèle. Ces deux fonctions

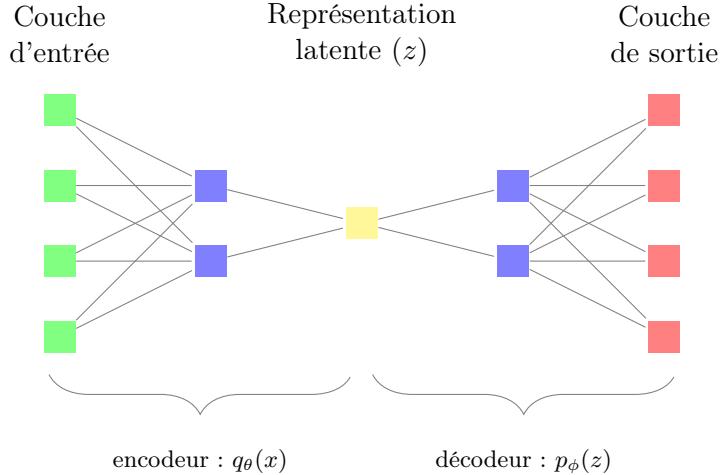


FIGURE 1.2 – Exemple illustrant la structure de base d'un autoencodeur. Dans le cas ci-dessus, on pourrait interpréter le schéma comme un réseau pleinement connecté où les blocs pourraient représenter les neurones et les liens seraient les poids, ou les paramètres du réseau. Le concept s'applique également à une architecture de réseau à convolutions, où les paramètres appris sont les filtres de convolutions.

vont permettre d'obtenir une sortie \hat{x} , donnée par l'équation :

$$\hat{x} = p_\phi\{q_\theta(x)\} \quad (1.6)$$

Ainsi, ce genre de méthode n'a pas besoin d'une étiquette y , car son objectif est basé sur x directement. C'est d'ailleurs pour cela qu'on parle d'une approche d'apprentissage non-supervisée. L'apprentissage des paramètres est fait en grande partie en minimisant l'erreur de reconstruction. La perte peut donc être définie par une fonction de la forme :

$$L(x, \hat{x}) = L(x, p_\phi\{q_\theta(x)\}) \quad (1.7)$$

où $q_\theta(\cdot)$ est l'encodeur et $p_\phi(\cdot)$ est le décodeur. L'optimisation de cette fonction de perte est faite par descente du gradient. En d'autres mots, les paramètres θ de l'encodeur et ϕ du décodeur sont optimisés graduellement en prenant la dérivée de la fonction de perte par rapport aux différents paramètres de ces deux composantes :

$$\Theta' \leftarrow \Theta - \epsilon * \frac{\partial L}{\partial \Theta} \quad (1.8)$$

où ϵ est un taux d'apprentissage qui permet de moduler la vitesse d'apprentissage et $\Theta = \{\theta, \phi\}$ comprend les paramètres de l'encodeur et du décodeur. Θ' correspond aux valeurs des paramètres après avoir appliqué la correction d'une itération.

Illustrons l'apprentissage d'un autoencodeur de base avec un exemple simpliste. Supposons que nous avons en entrée un jeu de données X avec $p = 2$ variables et $n = 10$ observations. Nous voulons encoder ces 2 variables dans une variable latente unidimensionnelle avec un autoencodeur à une seule couche cachée. Au total, notre autoencodeur possède 3 couches, soit une couche d'entrée, une couche cachée et une couche de sortie. Nous choisissons également une fonction d'activation sigmoïde pour notre couche cachée et une fonction d'activation linéaire pour notre couche de sortie. Les fonctions d'activation sont des transformations appliquées aux valeurs des neurones de notre réseau, ce qui permettra entre autres d'avoir un modèle non-linéaire, dans le cas de fonctions d'activation non-linéaires. L'architecture du réseau est définie dans la figure 1.3.

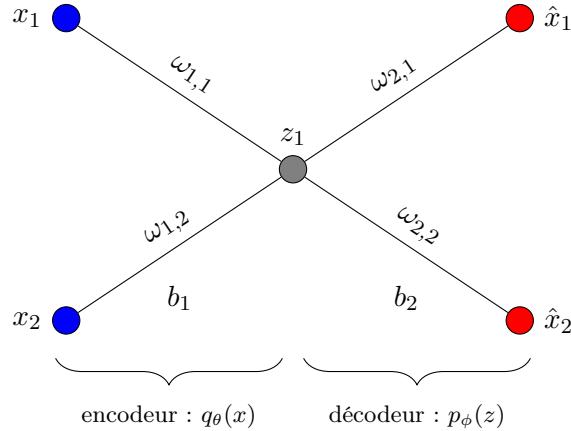


FIGURE 1.3 – Exemple simpliste d'une architecture d'autoencodeur

La fonction correspondant à l'encodeur, qui transforme l'entrée x vers la représentation latente z , est définie par l'équation 1.9. Dans cette équation, $h(\cdot)$ est la fonction d'activation sigmoïde donnée par : $h(x) = \frac{1}{1+e^{-x}}$. Les paramètres θ qui doivent être optimisés sont $w_{1,1}$, $w_{1,2}$ et b_1 .

$$q_\theta(x) = h(x_1 w_{1,1} + x_2 w_{1,2} + b_1) \quad (1.9)$$

Pour définir l'encodeur, il est également possible d'utiliser la notation matricielle définie par l'équation 1.10 où $X = [x_1, x_2]$. Pour une seule observation, on peut définir $X^{(i)} = [x_1^{(i)}, x_2^{(i)}]$. Le vecteur de poids \mathbf{w}_1 est définie comme $\mathbf{w}_1 = [w_{1,1}, w_{1,2}]$.

$$q_\theta(X) = h(X * \mathbf{w}_1 + b_1) \quad (1.10)$$

De la même manière que pour l'encodeur, il est possible de définir de manière plus détaillée la fonction associée au décodeur (équation 1.11).

$$p_\phi(Z) = \begin{cases} p_\phi^1(z) = z * w_{2,1} + b_2 \\ p_\phi^2(z) = z * w_{2,2} + b_2 \end{cases} \quad (1.11)$$

Les paramètres ϕ qui doivent être optimisés sont $w_{2,1}$, $w_{2,2}$ et b_2 . Encore une fois, il est possible d'utiliser une notation matricielle, comme définie à l'équation 1.12 où $\mathbf{w}_2 = [w_{2,1}, w_{2,2}]$:

$$p_\phi(Z) = h(Z * \mathbf{w}_2 + b_2) = \hat{X}, \quad p_\phi : \text{IR} \rightarrow \text{IR}^2. \quad (1.12)$$

Pour trouver les valeurs optimales des paramètres θ et ϕ du modèle, nous devons définir une fonction de perte (équation 1.7). Supposons que nous voulons minimiser l'erreur de reconstruction seulement. Nous pouvons alors définir notre fonction de perte de la manière suivante :

$$\begin{aligned} L(X, \hat{X}) &= L(X, p_\phi\{q_\theta(X)\}) \\ &= \frac{1}{2} \sum_{i=1}^n [x^{(i)} - p_\phi\{q_\theta(x^{(i)})\}]^T [x^{(i)} - p_\phi\{q_\theta(x^{(i)})\}] \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^p (x_j^{(i)} - p_\phi^j\{q_\theta(x_j^{(i)})\})^2 \end{aligned} \quad (1.13)$$

où i représente la i -ième observation et j représente le j -ième élément de l'entrée X . Maintenant que nous avons notre fonction de perte, nous pouvons optimiser les paramètres de nos fonctions encodeur et décodeur en utilisant la règle de dérivation en chaîne sur chacune des couches de notre réseau (voir équation 1.8).

Une fois que l'autoencodeur est entraîné, il est possible d'utiliser l'erreur de reconstruction comme score d'anomalie. En effet, une donnée mal reconstruite pourrait être vue comme une donnée aberrante, ou une donnée que le réseau n'a pas eu l'habitude de voir lors de l'entraînement. Dans Aggarwal (2016), ce genre de méthode de détection d'anomalies fait partie de la catégorie des algorithmes basés sur les modèles linéaires ou non-linéaires. Dans ce genre d'approche, on fait généralement l'apprentissage du modèle par l'erreur de reconstruction, où on cherche à minimiser les erreurs de prédiction. Toutefois, l'erreur de reconstruction n'est pas le seul critère qui peut être utilisé pour entraîner un autoencodeur. Dans la prochaine section, nous allons considérer un type précis d'autoencodeur, soit l'autoencodeur variationnel. Nous verrons également quelle autre composante peut être utilisée comme score d'anomalie.

1.3 Les autoencodeurs variationnels

Les autoencodeurs variationnels (VAE) Kingma and Welling (2013) ont une approche légèrement différente des autres autoencodeurs. En effet, au lieu d'encoder les données dans un vecteur de variables latentes à p dimensions, les données sont plutôt encodées dans 2 vecteurs de taille p : un vecteur de moyennes μ et un vecteur d'écart-types σ . Ces deux vecteurs sont ensuite utilisés comme paramètres d'une distribution paramétrique utilisée pour générer la représentation latente à p dimensions. Cette dernière représentation latente correspond donc dans ce cas-ci à une simulation d'une loi normale multivariée avec les paramètres μ et σ . Cela permet donc d'obtenir pour une donnée x , une représentation latente continue. C'est d'ailleurs la particularité la plus importante des VAE par rapport aux autres autencodeurs. Dans d'autres mots, les autoencodeurs de base ont comme objectif d'apprendre une représentation latente discrète, alors que les VAE apprennent une représentation continue qui représente plutôt une zone de cet espace latent. Cette zone est effectivement définie par les paramètres μ et σ associés à une entrée donnée. Cela veut aussi dire qu'une fois l'algorithme entraîné, la sortie de celui-ci est stochastique dans le sens où une entrée x peut donner 2 sorties différentes. Cependant, une donnée x va donner toujours les mêmes valeurs de μ et σ , cette composante n'est donc pas stochastique. La figure 1.4 illustre la structure de base des autoencodeurs variationnels. Dans la figure, on peut remarquer que les données en entrée commencent par passer par des couches cachées, qui peuvent être pleinement connectées ou de convolutions. Une couche pleinement connectée est une couche où chaque neurone est connecté avec tous les neurones de la couche suivante. Cette connexion est faite via un produit matriciel entre les valeurs des neurones et les poids du modèle. Une couche de convolutions est différente dans le sens où un filtre est appliqué à chaque sous région d'une matrice ou d'une image. Cette sous région est représentée par la dimensions du filtre appliqué. La convolution est fait en déplaçant le filtre avec un certain pas, que l'on appelle aussi *stride*. Lorsque le filtre est appliqué à une sous région, on obtient la valeur de la couche suivante en faisant le produit matriciel entre les valeurs de la sous région et les valeurs du filtre, qui représentent les paramètres du réseau. La première partie du réseau est l'encodeur ($q_\theta(x)$). Un peu avant d'arriver à la représentation latente, le réseau se divise en 2 composantes (μ et σ). La représentation latente z est ensuite générée en simulant d'une loi normale multivariée avec les paramètres μ et σ . C'est la fonction qu'on désigne par $q_\theta(z|x)$ dans la figure 1.4. Une fois la représentation z générée, celle-ci est décodée jusqu'au format original par des couches pleinement connectées ou des couches de déconvolutions. Pour les couches de déconvolutions, on fait l'opération inverse de la convolution. Cette partie est le décodeur ($p_\phi(z)$).

Les autoencodeurs variationnels sont également différents quant au calcul de perte qui est essentiel à l'optimisation du réseau. En effet, on ajoute une autre composante de perte à l'erreur de reconstruction de base. La fonction de perte est désormais définie comme une somme de deux composantes :

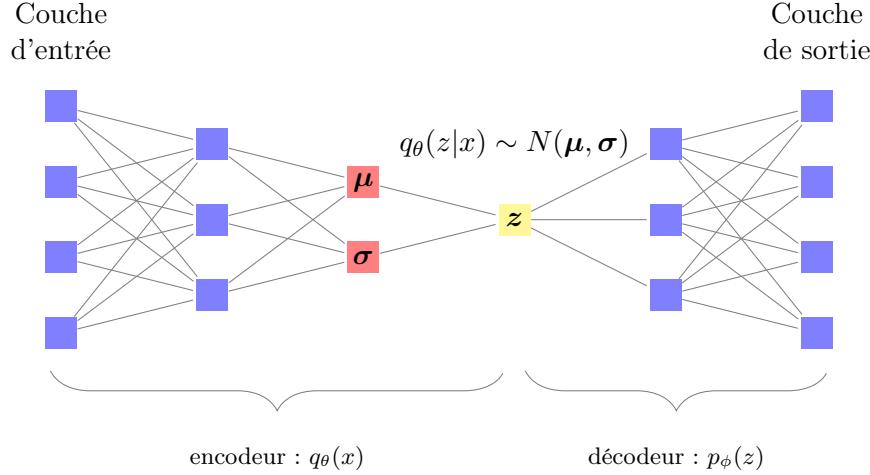


FIGURE 1.4 – Structure de base d'un autoencodeur variationnel. La représentation latente z est générée en simulant de la loi $q_\theta(z|x)$, qui suit une loi normale multivariée avec les paramètres μ et σ calculés par le réseau. C'est de là que vient la composante stochastique du modèle.

$$L(x, p_\phi\{q_\theta(x)\}) + D_{KL}[q_\theta(z|x)||p(z)], \quad (1.14)$$

où D_{KL} est une divergence de Kullback-Leibler. Cette mesure statistique permet de quantifier la différence entre 2 distributions. Pour 2 distributions continues A et B , la divergence de Kullbach-Leibler est donnée par :

$$\begin{aligned} D_{KL}(A||B) &= \int_{-\infty}^{\infty} a(x)\log\left(\frac{a(x)}{b(x)}\right) \\ &= \int_{-\infty}^{\infty} a(x)(\log(a(x)) - \log(b(x))). \end{aligned} \quad (1.15)$$

Dans le cas précis de l'autoencodeur variationnel, la distribution $q_\theta(z|x)$ est une loi normale multivariée de paramètres μ et σ . La loi $p(z)$, ou la loi à priori, est une loi normale multivariée standard $N(0, I)$, où I est la matrice identitaire. Dans le calcul de perte, plus la simulation provenant de la distribution $q_\theta(z|x)$ diverge d'une loi $N(0, I)$, plus la perte sera importante. Cela permet de restreindre la représentation latente dans un espace défini par la distribution à priori. Pour le calcul du critère de Kullbach-Leibler entre une loi $N(\mu, \sigma)$ et une loi $N(0, I)$ à k dimensions, l'équation 1.15 se développe comme suit :

$$\begin{aligned} D_{KL}(N_{\mu,\sigma}||N_{0,I}) &= \int_{-\infty}^{\infty} N_{\mu,\sigma}(x)\{\log(N_{\mu,\sigma}(x)) - \log(N_{0,I}(x))\} \\ &= \frac{1}{2}\{\text{tr}(\sigma^2) + \mu^T\mu - k - \log \det(\sigma^2)\}. \end{aligned} \quad (1.16)$$

La trace calculée sur σ^2 est la somme des variances sur les k dimensions. Étant donné que la matrice de variance-covariance est supposée diagonale, comme la fonction à priori, son déterminant se calcule comme le produit de sa diagonale. On peut donc simplifier l'équation 1.16 comme :

$$\begin{aligned} D_{\text{KL}}(N_{\mu, \sigma} || N_{0,I}) &= \frac{1}{2} \left\{ \sum_k \sigma_k^2 + \sum_k \mu_k^2 - \sum_k 1 - \log \prod_k \sigma_k^2 \right\} \\ &= \frac{1}{2} \left\{ \sum_k \sigma_k^2 + \sum_k \mu_k^2 - \sum_k 1 - \sum_k \log(\sigma_k^2) \right\} \\ &= \frac{1}{2} \sum_k \left\{ \sigma_k^2 + \mu_k^2 - 1 - \log(\sigma^2) \right\}. \end{aligned} \quad (1.17)$$

Pour des raisons de stabilité numérique, on utilise souvent en pratique la version logarithme de la matrice σ^2 (équation 1.17). Par contre, il est aussi possible de réécrire l'équation 1.17 sous cette forme :

$$D_{\text{KL}}(N_{\mu, \sigma} || N_{0,I}) = \frac{1}{2} \sum_k \left\{ \exp(\sigma_k^2) + \mu_k^2 - 1 - \sigma^2 \right\}. \quad (1.18)$$

L'objectif de cette nouvelle composante est de s'assurer que la distribution encodée $q_\theta(z|x)$ et la distribution à priori $p(z)$ sont similaires. Si cette composante de perte est suffisamment prise en compte lors de l'optimisation, nous devrions donc obtenir des paramètres μ et σ se rapprochant d'un vecteur de 0 et d'une matrice identitaire. Dans ce cas-ci, nous assumons l'indépendance entre les variables latentes, ce qui nous permet de définir σ comme un vecteur et non une matrice. Ce vecteur devient donc la diagonale de la matrice de variance-covariance.

Comme expliquée un peu plus tôt, la représentation latente de ce type d'autoencodeur est simulée d'une loi normale multivariée, donc stochastique. Cela pourrait en théorie rendre complexe la tâche d'optimisation du réseau. En effet, les paramètres du réseau, soit ceux de l'encodeur et du décodeur, sont optimisés par descente du gradient. On calcule donc la perte $L(x)$ et on dérive cette fonction par rapport à chaque paramètres du réseau. Mais qu'en est-il de la loi normale multivariée qui a généré notre représentation latente ? Afin de simplifier le calcul des dérivées lors de la rétropropagation, on redéfinit le calcul de la représentation z dans un format plus simple. Cette simplification est communément appelée la *reparametrization trick*. En effet, il est possible pour certaines distributions, comme la loi normale, de séparer les paramètres (μ et σ) de la composante stochastique. Concrètement, on peut définir une simulation normale multivariée comme :

$$z = \mu + \sigma * \epsilon$$

où $\epsilon \sim N(0, I)$. En bref, cela signifie que la couche z , illustrée dans la figure 1.4, est générée à partir de deux couches de paramètres μ et σ et d'une simulation $N(0, I)$. Cette réécriture permet d'isoler la composante stochastique associée à la loi normale. Lors de la rétropropagation, on peut calculer les dérivées des paramètres μ et σ seulement et ignorer ϵ .

1.3.1 β -VAE

La composante de perte associée au critère de divergence de Kullbach-Leibler apporte de la régularisation au modèle en restreignant celui-ci dans un certain espace (Kingma and Welling, 2013). Cette régularisation est d'autant plus importante dans un contexte où les réseaux de neurones ont un potentiel d'apprentissage important et où il devient primordial d'éviter le sur-apprentissage. Par contre, trop de régularisation pourrait amener un réseau à sous-apprendre. Il est donc important de trouver le bon équilibre entre la perte associée à la reconstruction et la perte associée au critère de Kullbach-Leibler.

Pour trouver ce bon équilibre, on peut ajouter un hyperparamètre à notre fonction de perte. Cet hyperparamètre, défini par β , permet de donner plus ou moins d'importance au critère de KL. Avec cet hyperparamètre, la fonction de perte est maintenant donnée par :

$$L(x, p_\phi\{q_\theta(x)\}) + \beta \times D_{KL}[q_\theta(z|x) || p(z)]. \quad (1.19)$$

Dans ce cas-ci, on parle plutôt d'un β -VAE, qui a été introduit par Higgins et al. (2017). Plus cet hyperparamètre β est élevé, plus la régularisation est importante et aussi plus les éléments de la représentation latente seront dissociés (ce qu'on appelle une représentation *disentangled* en anglais). Cela est dû à l'invariance de la fonction à priori $p(z)$ sur laquelle ce critère de perte est basé, soit une loi normale multivariée avec moyenne nulle et covariance σI . Cette propriété de *disentanglement* peut devenir intéressante dans le cas où on souhaite avoir des variables latentes indépendantes qui expliquent différents aspects des données. Un exemple cité dans leur article fait référence à un jeu de données synthétiques de visages. Après avoir entraîné un réseau qui permet d'obtenir des variables latentes *disentangled*, ils sont capables de démontrer que chaque variable latente à une fonction précise dans l'image reconstruite : rotation, éclairage, élévation, etc. Il est possible d'observer ces fonctions en faisant varier une variable latente seulement.

1.3.2 La détection d'anomalies avec un VAE

Pour ce qui est de la détection d'anomalies, les autoencodeurs variationnels, ou les β -VAE, peuvent en théorie être utilisés de la même manière qu'un autoencodeur de base. En effet, le VAE calcule une erreur de reconstruction qui peut ensuite être utilisée comme score d'anomalie. Par contre, une autre composante du VAE peut potentiellement être intéressante dans un

contexte de détection d'anomalie. C'est d'ailleurs autour de cela que tourne le sujet de ce mémoire. Il s'agit de la représentation latente qui est basée sur une distribution à priori. Étant donné que la fonction de perte du VAE pénalise une représentation latente s'éloignant de cette distribution à priori, il intéressant de voir comment se comporte cette représentation latente pour des données normales comparativement à des anomalies. Cette représentation, potentiellement riche en information, possède généralement un niveau de complexité bien inférieur à l'entrée. En bref, le fait d'obtenir une représentation simple et riche en information sur laquelle on a un à priori pourrait devenir une possibilité intéressante quant à la détection d'anomalies. C'est d'ailleurs un sujet que nous reverrons plus loin dans la description de la méthodologie.

Avec ce survol des méthodes de détection d'anomalies et des autoencodeurs, nous sommes en mesure de décrire plus en détails notre approche proposée.

Chapitre 2

Méthodologie

Le cœur de notre travail consiste à proposer une méthode de détection d'anomalies où nous utilisons la représentation latente d'un autoencodeur variationnel. Cette représentation servira ensuite comme base pour notre cadre décisionnel nous permettant de détecter des anomalies dans un jeu de données.

2.1 Les objectifs de l'approche

Le premier objectif de notre approche est de faire la détection d'anomalies sur des données complexes, comme par exemple des images. En second lieu, nous souhaitons avoir une approche qui discrimine des anomalies en utilisant un seuil qui s'apparente à un niveau de confiance plutôt qu'une métrique quelconque qui peut être difficile à établir et à interpréter. Sachant ces 2 objectifs, il faut que notre méthodologie soit en mesure de traiter des données complexes et doit nous donner en sortie un score d'anomalie facile à interpréter.

Tout d'abord, il est pertinent de mentionner que le choix d'utiliser des réseaux de neurones, plus particulièrement des autoencodeurs, est étroitement lié au fait de traiter des données complexes. En effet, les réseaux de neurones ont le potentiel d'apprendre des relations complexes et non-linéaires. De plus, l'architecture du réseau peut être adaptée selon la complexité des données en ajoutant des couches de paramètres. Finalement, les réseaux à convolutions sont également bien adaptés au domaine de l'imagerie en prenant en compte l'information de manière locale dans une image.

Ensuite, l'avantage d'avoir un score d'anomalie facile à interpréter en sortie est de simplifier la prise de décision pour discriminer les anomalies des observations normales. Pour illustrer cet avantage, prenons un contre-exemple où l'on veut faire la détection d'anomalies à partir d'une méthode basée sur une distance comme les k plus proches voisins. Dans cette approche, nous pourrions considérer comme anormales les observations pour lesquelles il y a moins de k voisins à l'intérieur d'un rayon de longueur ϵ . La détection d'anomalie requiert donc de

définir les paramètres k et ϵ , qui ne pourront pas être choisis aussi facilement qu'un niveau de confiance. Un score d'anomalie que l'on pourrait interpréter de manière similaire à un niveau de confiance α serait effectivement beaucoup plus simple et objectif à déterminer.

2.2 Les hypothèses de l'approche

Dans notre approche, nous supposons que nous avons accès à un jeu de données d'entraînement $\mathcal{X} = \{\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(n)}\}$ qui contient n observations indépendantes de $\mathbf{X} \in \mathbb{R}^{d_1 \times d_2}$. Dans notre cas, on suppose que les matrices sont carrées, soit $d_1 = d_2 = d$. Pour simplifier la notation, ces matrices peuvent être exprimées comme des vecteurs de longueur d^2 . On peut donc réécrire sous la forme vectorielle $\mathbf{x}^{(i)} = [x_1^{(i)}, \dots, x_{d^2}^{(i)}]$; la façon d'obtenir cette forme vectorielle a peu d'importance, tant qu'on procède toujours de la même manière pour toutes les observations. Ces représentations peuvent être obtenues par ligne ou par colonne. Dans notre cas, nous avons dû procéder à ce genre de transformation pour l'une des approches comparatives testées dans nos applications, soit l'analyse en composantes principales à noyau. Pour ce faire, nous avons aplati les matrices (images) en vecteurs. La figure 2.1 illustre comment nous avons procédé pour les images à un seul canal et les images de type *RGB*.

Les n observations de notre jeu d'entraînement proviennent d'un mélange à proportion $(1 - p)$ d'observations identiquement distribuées dites "normales" et p d'observations identiquement distribuées dites anomalies. Pour indiquer si la matrice $\mathbf{X}^{(i)}$ provient de la population dite "normale" \mathcal{N} , soit $\mathbf{X}^{(i)} \in \mathcal{N}$, on utilise la fonction indicatrice définie ci-dessous,

$$\delta_i = \begin{cases} 0 & \text{si } \mathbf{X}^{(i)} \in \mathcal{N} \\ 1 & \text{si } \mathbf{X}^{(i)} \notin \mathcal{N}. \end{cases} \quad (2.1)$$

On suppose que la proportion p est faible, typiquement moins de 5%. Dans la pratique, nous ne connaissons pas la valeur des δ_i , ce qui veut dire que nous ne pouvons connaître le pourcentage réel d'anomalies p .

Nous avons également k observations indépendantes de matrices constituées des mêmes $d \times d$ dimensions dans un jeu de données test $\mathcal{X}^* = \{\mathbf{X}^{*(1)}, \dots, \mathbf{X}^{*(k)}\}$. On suppose que le pourcentage d'anomalies p^* dans ce jeu de données est similaire ou supérieur à p . C'est donc dire que parmi ces k observations, $(1 - p^*)$ proviennent de la même distribution \mathcal{N} que les données du jeu d'entraînement et p^* sont des anomalies (également de la même population d'anomalies que le jeu d'entraînement). De la même manière que pour le jeu de données d'entraînement \mathcal{X} , nous ne savons pas si une matrice \mathbf{X}^* du jeu de données test provient de la population dite "normale" \mathcal{N} . Nous ne connaissons donc pas non plus le pourcentage réel p^* . Pour indiquer si la matrice $\mathbf{X}^{*(i)}$ provient de la population dite "normale" \mathcal{N} , soit $\mathbf{X}^{*(i)} \in \mathcal{N}$, on utilise la même fonction indicatrice que celle définie dans l'équation 2.2 :

Image brute	Version matricielle	Version vectorielle																										
	<table border="1"> <tr><td>2</td><td>5</td><td>3</td><td>10</td></tr> <tr><td>5</td><td>90</td><td>150</td><td>95</td></tr> <tr><td>4</td><td>10</td><td>120</td><td>75</td></tr> <tr><td>6</td><td>50</td><td>130</td><td>60</td></tr> <tr><td>45</td><td>155</td><td>55</td><td>20</td></tr> </table>	2	5	3	10	5	90	150	95	4	10	120	75	6	50	130	60	45	155	55	20	<table border="1"> <tr><td>2</td><td>5</td><td>3</td><td>...</td><td>55</td><td>20</td></tr> </table>	2	5	3	...	55	20
2	5	3	10																									
5	90	150	95																									
4	10	120	75																									
6	50	130	60																									
45	155	55	20																									
2	5	3	...	55	20																							

(a) Images à un seul canal ("noir et blanc")

Image brute	Version matricielle	Version vectorielle																																								
	<table border="1"> <tr><td>32</td><td>9</td><td>30</td><td>121</td></tr> <tr><td>78</td><td>16</td><td>16</td><td>35</td></tr> <tr><td>2</td><td>5</td><td>3</td><td>10</td></tr> <tr><td>5</td><td>90</td><td>150</td><td>95</td></tr> <tr><td>4</td><td>10</td><td>120</td><td>75</td></tr> <tr><td>6</td><td>50</td><td>130</td><td>60</td></tr> <tr><td>45</td><td>155</td><td>55</td><td>20</td></tr> </table>	32	9	30	121	78	16	16	35	2	5	3	10	5	90	150	95	4	10	120	75	6	50	130	60	45	155	55	20	<table border="1"> <tr><td>2</td><td>5</td><td>...</td><td>20</td><td>78</td><td>16</td><td>...</td><td>98</td><td>32</td><td>9</td><td>...</td><td>43</td></tr> </table>	2	5	...	20	78	16	...	98	32	9	...	43
32	9	30	121																																							
78	16	16	35																																							
2	5	3	10																																							
5	90	150	95																																							
4	10	120	75																																							
6	50	130	60																																							
45	155	55	20																																							
2	5	...	20	78	16	...	98	32	9	...	43																															

(b) Images à 3 canaux (*RGB*)

FIGURE 2.1 – Figure illustrant notre méthodologie pour transformer une image d’entrée brute en format vectoriel selon le nombre de canaux de l’image. Les valeurs présentées dans les matrices sont fictives.

$$\delta_i^* = \begin{cases} 0 & \text{si } \mathbf{X}^{*(i)} \in \mathcal{N} \\ 1 & \text{si } \mathbf{X}^{*(i)} \notin \mathcal{N}. \end{cases} \quad (2.2)$$

2.3 Description de l’approche

Notre approche se divise essentiellement en 2 étapes. La première étape implique d’entraîner un autoencodeur variationnel pour apprendre les caractéristiques de la population normale du jeu de données d’entraînement \mathcal{X} . La deuxième étape consiste à définir un cadre décisionnel à

partir des représentations latentes de l'autoencodeur entraîné nous permettant ainsi d'identifier les anomalies dans le jeu de données de test \mathcal{X}^* avec seuil α similaire à un niveau de confiance, que nous appellerons niveau de filtration.

2.3.1 Entrainer l'autoencodeur

La première étape consiste à utiliser le jeu de données d'entraînement \mathcal{X} pour entraîner l'autoencodeur variationnel. Étant donné que \mathcal{X} ne contient presque pas d'anomalies, cela devrait permettre d'apprendre les caractéristiques, ou la distribution, de la population dite "normale". Cette distribution apprise sera contenue dans la représentation latente du VAE entraîné, ou plus spécifiquement dans les couches μ et σ du réseau. Ces deux couches précèdent la couche latente et permettent de générer celle-ci de manière stochastique, comme nous l'avons vu dans la section 1.3. Les VAE sont généralement entraînés avec une fonction de coût à deux composantes (voir l'équation 1.14). Une de ces deux composantes est associée à la représentation latente du réseau, s'assurant ainsi que celle-ci s'approche d'une distribution a priori, soit une $N(0, I)$ dans notre cas. Pour ce faire, les valeurs des couches μ et σ doivent s'approcher des paramètres de la loi a priori, soit les vecteurs $(\mathbf{0}, \mathbf{1}) = (0^m, 1^m)$ dans le cas d'une représentation latente à m dimensions ; nous pourrons d'ailleurs tirer avantage de cette hypothèse a priori dans notre règle de décision que nous allons décrire en détails à la prochaine section. La deuxième composante de la fonction de perte, soit l'erreur de reconstruction, demeure primordiale dans l'entraînement du réseau. Dans sa forme la plus simple, cette erreur de reconstruction est définie pixel par pixel. Dans le cas de l'erreur quadratique moyenne, cette composante de perte peut être définie pour la contribution d'une observation i comme

$$L(\mathbf{x}^{(i)}, p_\phi\{q_\theta(\mathbf{x}^{(i)})\}) = \frac{1}{d^2} \sum_{l=1}^{d^2} (x_l^{(i)} - p_\phi\{q_\theta(\mathbf{x}^{(i)})\}_l)^2, \quad (2.3)$$

où $p_\phi\{q_\theta(\mathbf{x}^{(i)})\}_l$ est le l -ème élément du vecteur ou de la matrice de sortie $p_\phi\{q_\theta(\mathbf{x}^{(i)})\}$ donnée par l'équation 1.6. Dans le cas d'une image, le l -ème élément correspond au l -ème pixel de l'image.

La fonction de perte définie à l'équation 2.3 accorde autant d'importance à chacun des pixels de l'image. C'est donc dire que le réseau a pour objectif de bien reconstruire autant les pixels en arrière-plan que les pixels centraux. Dans notre cas, on cherche à trouver des anomalies au niveau du contenu global de l'image. Pour illustrer ce qu'on veut dire par contenu global, la figure 2.2 montre quelques exemples d'images considérées comme "normales" et "anormales". Dans la figure 2.2, on peut y voir que les images "normales" de la sous-figure 2.2a correspondent à des voitures dans un environnement extérieur. La sous-figure 2.2b présente des images "anormales", où on peut y apercevoir des chiens dans un environnement extérieur. Même si l'arrière-plan de ces images est similaire à celui de 2.2a, le contenu global correspond

à des images de chiens et non de voiture. À l'opposé, on peut apercevoir à la sous-figure 2.2c des images de voitures correspondant à des modèles particuliers, des environnements intérieurs et aussi contenant des écritures sur les marges supérieures et inférieures de l'image. Malgré ces observations, le contenu global de ces images correspond tout de même à des voitures, ce qui nous amène à les considérer comme des images dites "normales".

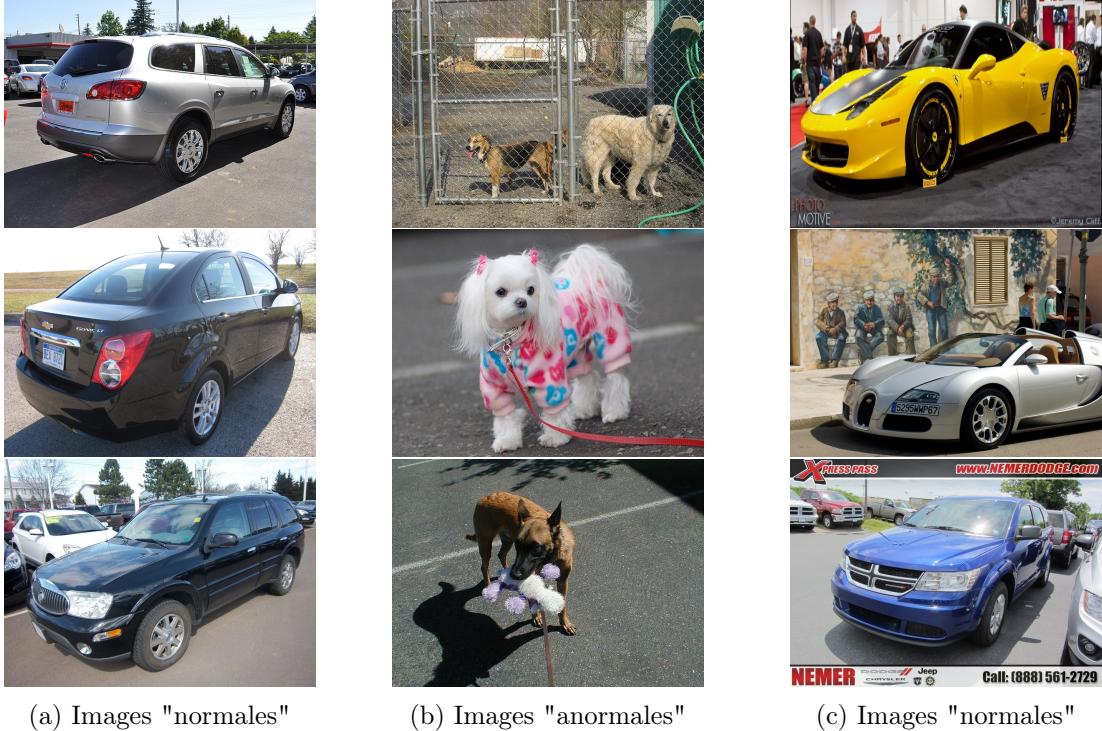


FIGURE 2.2 – Figure illustrant des exemples d’images considérées comme "normales" et d’autres comme "anormales". Dans ce cas, les images dites "normales" sont des images de voitures.

La figure 2.2 illustre le fait qu'il devient pertinent de trouver une manière d'accorder plus d'importance au contenu global de l'image. Pour ce faire, on utilise un concept qui s'appelle *perceptual loss*. Dans ce type d'optimisation, on calcule la perte en se basant sur une couche spécifique d'un autre réseau de neurones pré-entraîné sur beaucoup plus d'images. En pratique, on utilise généralement des architectures de réseaux connus, comme *VGG* (Simonyan and Zisserman, 2014) ou *ResNet* (He et al., 2015), pré-entraînés sur *ImageNet* (Deng et al., 2009). Cette approche est d'ailleurs utilisée pour transformer le style d'une image, par exemple pour donner le style d'un peintre célèbre à une image quelconque (Johnson et al., 2016). La figure 2.3 illustre le mécanisme exact du calcul de la perte. On peut y voir que l'erreur de reconstruction du réseau n'est plus calculée directement entre la sortie du réseau \hat{x} et l'entrée x . En effet, les deux composantes sont plutôt données en entrée à un autre réseau, et l'erreur est calculée en comparant plutôt une représentation précise de ce réseau obtenue par ces 2 composantes.

Étant donné que le réseau correspondant à la fonction $g(a)$ dans la figure 2.3 est pré-entraîné

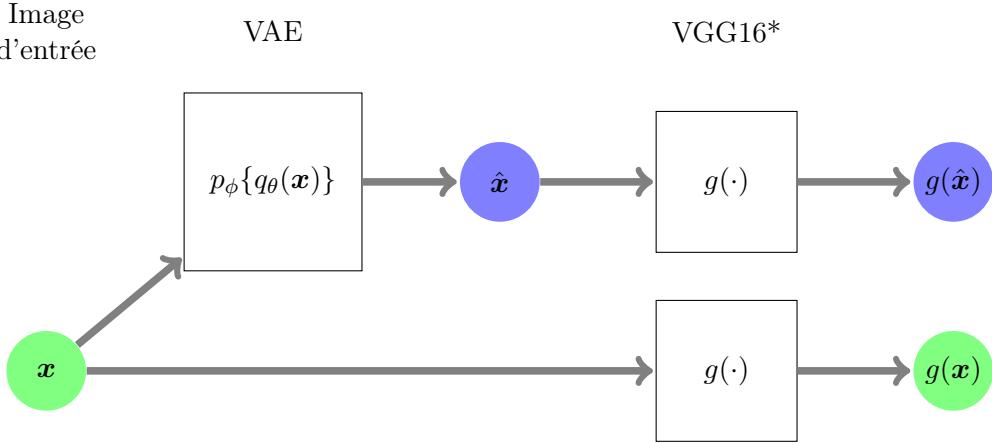


FIGURE 2.3 – Figure montrant le mécanisme derrière le concept de *perceptual optimization*. Au lieu de calculer la perte entre \mathbf{x} et $\hat{\mathbf{x}}$, la perte est calculée entre $g(\mathbf{x})$ et $g(\hat{\mathbf{x}})$. La fonction $g(a)$ permet d'extraire les valeurs d'une couche spécifique d'un autre réseau pré-entraîné, comme par exemple *VGG16* pré-entraîné sur *ImageNet*.

sur plusieurs images réelles, cela nous permet d'extraire une couche qui contient de l'information à plus bas niveau comme la reconnaissance de lignes, des formes précises, des couleurs, etc. Plus la couche sélectionnée est près de l'entrée du réseau, plus l'information utilisée pour calculer la perte sera simple et brute. À l'inverse, si on sélectionne une couche plus près de la sortie, nous serons en mesure de prendre en compte de l'information plus complexe, par exemples des formes particulières (Johnson et al., 2016). L'erreur de reconstruction qui était définie à l'équation 2.3, peut maintenant se définir comme suit :

$$L(\mathbf{x}^{(i)}, p_\phi\{q_\theta(\mathbf{x}^{(i)})\}) = \frac{1}{d^2} \sum_{l=1}^{d^2} \left[g(\mathbf{x}^{(i)})_l - g(p_\phi\{q_\theta(\mathbf{x}^{(i)})\})_l \right]^2. \quad (2.4)$$

L'objectif d'utiliser ce type de perte est d'optimiser la reconstruction du réseau non pas sur les valeurs précises des pixels reconstruites en sortie, mais plutôt sur des représentations contenant de l'information plus structurée. Cela permet entre autres d'orienter l'apprentissage vers la reconstruction du contenu à plus haut-niveau, plutôt que sur la reconstruction parfaite de chaque pixel. Il est important de rappeler que notre objectif est de trouver des anomalies, et non reconstruire parfaitement des images.

Une fois l'autoencodeur variationnel entraîné, c'est-à-dire lorsque les paramètres θ et ϕ sont fixés, il nous est donc possible d'utiliser la partie encodeur du réseau, $q_\theta(\cdot)$, pour transformer les images, ou les données en entrée, en une paire de vecteurs $(\boldsymbol{\mu}, \boldsymbol{\sigma})$:

$$q_\theta(\mathbf{x}) = (\boldsymbol{\mu}, \boldsymbol{\sigma}), \quad \boldsymbol{\mu} \in \mathbb{R}^m, \boldsymbol{\sigma} \in \mathbb{R}^m. \quad (2.5)$$

Certains détails supplémentaires concernant l'entraînement de l'autoencodeur variationnel seront décrits dans la section 3.2.1.

2.3.2 Définir le cadre décisionnel

La prochaine étape de notre approche consiste à définir un cadre décisionnel nous permettant de discriminer les anomalies des observations "normales". Avec ce cadre décisionnel, on cherche à savoir si une observation provenant du jeu de données test, soit \mathbf{X}^* , provient de la population normale \mathcal{N} . Pour ce faire, nous allons définir une métrique ou une statistique de distance, obtenue via une fonction $T(\cdot)$, que nous pourrons ensuite comparer à un seuil de décision s . La valeur de la statistique de distance et le seuil nous permettront de définir une région $R_N = \{x : T(x) < s\}$ ou $R_N = \{x : T(x) > s\}$, pour un $s > 0$. Si une observation $\mathbf{x}^{(i)}$ se retrouve dans cette région R_N , nous pourrons supposer que celle-ci est de la population \mathcal{N} , donc poser $\delta_i^* = 0$. Afin de prendre en compte l'apprentissage fait par l'autoencodeur variationnel, la statistique de distance sera plutôt basée sur la représentation latente obtenue par l'encodeur $q_\theta(\cdot)$ pour une observation $\mathbf{x}^{(i)}$. Cela nous permet de réécrire la région R_N :

$$R_N = \{x : T(q_\theta(\mathbf{x}^{(i)})) < s\}. \quad (2.6)$$

Pour obtenir cette région R_N , il faut donc procéder en 2 étapes. La première consiste à calculer la statistique de distance, donnée par la fonction $T(\cdot)$, sur chacune des instances de notre jeu de données d'entraînement. La deuxième étape consiste à calculer le seuil s avec l'aide d'un niveau de filtration α similaire à un niveau de confiance. Ces deux étapes nous permettront de définir notre région de décision nous permettant d'inférer quelles observations du jeu de données test devraient être étiquetées comme des anomalies. Les prochaines sections décrivent plus en détails comment calculer les statistiques de distance et comment définir le seuil à partir de celles-ci.

Définir une statistique de distance

À partir de la section 2.3.1, nous sommes en mesure d'obtenir des représentations latentes, encodées sous forme de vecteurs, pour chaque instance de notre jeu de données \mathcal{X} . Nous allons utiliser ces représentations latentes pour générer une statistique de distance selon laquelle nous prévoyons un comportement différent selon si une observation est "normale" ou "anormale". La métrique que nous avons choisie pour définir notre statistique de distance est la distance de Kullbach-Leibler entre une loi $N(\boldsymbol{\mu}, \boldsymbol{\sigma})$ et une loi $N(0, I)$, où les vecteurs $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ correspondent à la représentation latente obtenue par l'autoencodeur variationnel. Cette statistique de distance, que nous appellerons T , est donc définie comme suit pour une observation i :

$$T^{(i)} = D_{KL}[N(\boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{(i)}) || N(0, I)]. \quad (2.7)$$

La statistique de distance T nous permet donc de quantifier la distance entre les vecteurs $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ de la représentation latente d'une observation i et les vecteurs $(\mathbf{0}_m, \mathbf{1}_m)$ correspondant aux paramètres d'une loi $N(0, I)$. Nous pouvons donc calculer cette distance pour chacune des n observations du jeu de données d'entraînement \mathcal{X} pour obtenir l'ensemble de valeurs $T_{\mathcal{X}} = \{T^{(1)}, \dots, T^{(n)}\}$. Le raisonnement derrière le choix de cette métrique réside dans le fait que celle-ci est utilisée comme composante dans le critère de perte utilisé lors de l'entraînement de l'autoencodeur. Si l'importance relative de cette composante de perte est différente entre les anomalies et les observations "normales", cette différence devrait se refléter dans les représentations latentes, donc aussi dans la statistique de distance calculée. Le débordement naturel des anomalies dans un jeu de données nous amène à faire l'hypothèse que les deux composantes de perte ne seront pas optimisées de manière similaire entre les anomalies et les observations "normales".

Maintenant que nous avons fait l'hypothèse que les valeurs des statistiques de distance des anomalies différeront des valeurs des statistiques de distance des observations "normales", il reste à savoir si une valeur élevée est un signe d'anomalie ou l'inverse. Pour savoir comment interpréter la valeur de la statistique, il faut inspecter certaines observations du jeu de données d'entraînement. Nous proposons d'analyser certaines observations ayant des statistiques de distance élevées et certaines observations ayant des statistiques de distance faibles. En analysant les deux extrémités, nous serons en mesure de conclure comment les représentations latentes des observations dites "normales" se comportent par rapport à la loi $N(0, I)$. Nos expérimentations révèlent 2 scénarios possibles par rapport au comportement des représentations latentes observations "normales" par rapport à la $N(0, I)$:

1. **Près de la $N(0, I)$** : Les valeurs des statistiques de distance correspondant aux observations "normales" sont faibles dans l'ensemble $T_{\mathcal{X}}$.
2. **Éloigné de la $N(0, I)$** : Les valeurs des statistiques de distance correspondant aux observations "normales" sont élevées dans l'ensemble $T_{\mathcal{X}}$.

Les deux scénarios décrits ci-dessus sont illustrés en exemple dans la figure 2.4. Dans cette figure, on peut voir le paramètre μ sur l'axe des x et le paramètre σ^2 sur l'axe des y , ce qui représenterait le cas où les représentations latentes auraient 1 seule dimension, soit une valeur générée par une loi $N(\mu, \sigma^2)$. On peut y voir la différence par rapport au point central (les 2 croix noires) entre les observations connues "normales" (les points rouges) et les observations connues "anormales" (les points bleus). La croix en forme de "+" correspond au point central des observations "normales" et la croix en forme de "x" correspond au point central des observations "anormales". Cette figure illustre le cas où l'on connaît l'étiquette de tous les

observations. Dans un cas réel, nous ne connaîtrions pas ces étiquettes, il faudra donc tenter de déterminer lequel de ces 2 scénarios semble le plus plausible en analysant certaines observations étant très près et très éloignées du point central ($\mathbf{0}_m$, $\mathbf{1}_m$).

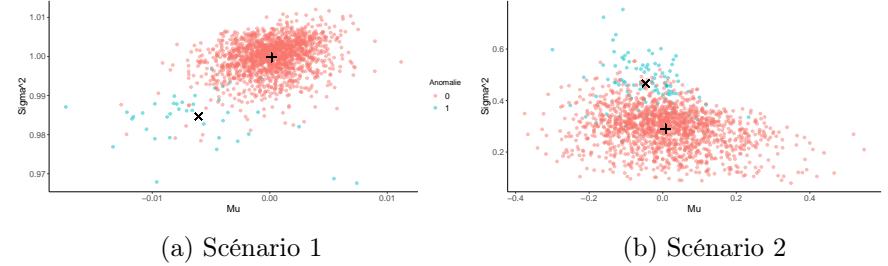


FIGURE 2.4 – Exemples de situations représentant les 2 différents scénarios se rapportant aux représentation latentes des observations "normales" et "anormales".

Une fois que nous avons défini et analysé les statistiques de distance obtenues sur l'ensemble d'entraînement, nous pouvons passer à la prochaine étape qui consiste à trouver le seuil s nous permettant de définir la région de décision R_N .

Trouver le seuil s

Pour trouver le seuil s , on veut partir d'un niveau de filtration α similaire à un niveau de confiance. Ce niveau de filtration α nous permettra de faciliter la détermination du seuil s . Le seuil déterminé avec un niveau de filtration α sera appelé $s_{(1-\alpha)}$. Par exemple, avec un niveau de filtration $\alpha = 0.05$, notre seuil s nous permettra d'identifier les observations ayant des statistiques de distances se retrouvant dans les 95% plus "normales" de l'ensemble d'entraînement \mathcal{X} . En faisant varier ce paramètre α , nous pourrons identifier plus ou moins d'anomalies.

Pour fixer la exacte du seuil $s_{(1-\alpha)}$, nous allons extraire une valeur de notre ensemble de statistiques de distance $T_{\mathcal{X}}$. Nous pourrons ensuite comparer cette valeur $s_{(1-\alpha)}$ à la valeur d'une statistique de distance correspondant à une nouvelle observation pour conclure si cette nouvelle observation fait partie de la région R_N . Le scénario identifié à la section précédente nous est nécessaire pour trouver la valeur du seuil $s_{(1-\alpha)}$. Dans le scénario où les observations "normales" sont près de la $N(0, I)$, soit le scénario 1, nous allons préalablement ordonner les valeurs de $T_{\mathcal{X}}$ en ordre croissant. Dans le scénario inverse où les observations "normales" sont éloignées de la $N(0, I)$, soit le scénario 2, nous allons plutôt ordonner les valeurs de $T_{\mathcal{X}}$ en ordre décroissant. Nous définissons $T'_{\mathcal{X}}$, l'ensemble $T_{\mathcal{X}}$ ordonné selon le scénario en question. Le seuil $s_{(1-\alpha)}$ correspond donc à la valeur associée au rang $\lfloor (1 - \alpha) \times n \rfloor$ de l'ensemble $T'_{\mathcal{X}}$:

$$s_{(1-\alpha)} = T'_{\mathcal{X}\{(1-\alpha)\times n\}}, \quad (2.8)$$

où $T'_{\mathcal{X}\{(1-\alpha) \times n\}}$ est la valeur correspondant au rang $\lfloor(1-\alpha) \times n\rfloor$ de l'ensemble $T'_{\mathcal{X}}$. Ce seuil nous permet de définir la région R_N , qui représente la région où une observation est considérée comme "normale". Si on définit t comme étant la valeur d'une statistique de distance calculée comme $t = T(q_\theta(\mathbf{x}))$, la région R_N est donnée par :

$$R_N = \begin{cases} \{t \in \mathbb{R} : t < s_{(1-\alpha)}\} & \text{si les observations "normales" sont près de } N(0, I) \\ \{t \in \mathbb{R} : t > s_{(1-\alpha)}\} & \text{si les observations "normales" sont éloignées de } N(0, I) \end{cases} \quad (2.9)$$

Pour tester les observations de notre jeu de données test, nous allons calculer les statistiques de distance T des k observations de ce jeu de données \mathcal{X}^* . Si la valeur de la statistique de distance $T^{(j)}$ correspondant à l'observation $\mathbf{x}^{*(j)}$ fait partie de la région R_N , on peut donc conclure que l'observation fait partie de l'ensemble $(1-\alpha)$ -normal. Si la statistique de distance $T^{(j)}$ ne fait pas partie de la région R_N , on l'étiquettera comme une anomalie.

La région R_N nous permet de définir si une observation est une α -anomalie ou non. De manière similaire, on peut également utiliser l'ensemble $T_{\mathcal{X}}$ pour établir un score d'anomalie pour une observation donnée. Ce score d'anomalie est défini entre $[0, 1]$, où une anomalie évidente devrait avoir un score se rapprochant de 1. Pour établir ce score, il suffit de calculer la statistique de distance $T^{(j)}$ correspondant à l'observation $\mathbf{x}^{*(j)}$, ensuite de calculer le rang de cette statistique de distance parmi l'ensemble $T'_{\mathcal{X}}$ et finalement diviser ce rang par le nombre d'observations dans l'ensemble $T'_{\mathcal{X}}$, soit n dans notre cas. Ce score d'anomalie, défini comme γ , peut être obtenu pour une observation $\mathbf{x}^{*(j)}$:

$$\gamma(\mathbf{x}^{*(j)}) = \frac{\text{rang}_{T'_{\mathcal{X}}}(T^{(j)})}{n} \quad (2.10)$$

où $\text{rang}_{T'_{\mathcal{X}}}(T^{(j)})$ correspond au rang de la statistique de distance $T^{(j)}$ dans l'ensemble ordonné $T'_{\mathcal{X}}$. Une valeur élevée du score défini à l'équation 2.10, ou près de 1, signifie que l'observation $\mathbf{x}^{*(j)}$ est considérée comme une anomalie selon l'algorithme.

L'algorithme 1 résume les différentes étapes de notre approche.

Algorithme 1 : Algorithme de détection d'anomalies basé sur les représentations latentes d'un autoencodeur variationnel

Input : Ensemble de données d'entraînement avec une proportion p d'anomalies

$\mathbf{x}^{(i)}, i = 1, \dots, n$

Ensemble de données de test avec anomalies $\mathbf{x}^{*(j)}, j = 1, \dots, k$

Niveau de filtration α

Output : Indicateurs d'anomalies $o^{(j)}, j = 1, \dots, k$

$\theta, \phi \leftarrow$ paramètres de l'encodeur ($q_\theta(\cdot)$) et du décodeur ($p_\phi(\cdot)$) du VAE entraîné;

for $i=1$ to n **do**

$(\boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{(i)}) = q_\theta(\mathbf{x}^{(i)})$

$T_{\mathcal{X}}^{(i)} = D_{KL}[N(\boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{(i)}) || N(0, I)]$

end

Ordonner $T_{\mathcal{X}}$ selon le scénario identifié pour obtenir $T'_{\mathcal{X}}$;

for $j=1$ to k **do**

$(\boldsymbol{\mu}^{(j)}, \boldsymbol{\sigma}^{(j)}) = q_\theta(\mathbf{x}^{*(j)})$

$T_{\mathcal{X}^*}^{(j)} = D_{KL}[N(\boldsymbol{\mu}^{(j)}, \boldsymbol{\sigma}^{(j)}) || N(0, I)]$

$\gamma(\mathbf{x}^{*(j)}) = \text{rang}_{T'_{\mathcal{X}}} (T_{\mathcal{X}^*}^{(j)}) / n$

if $\gamma(\mathbf{x}^{*(j)}) \geq (1 - \alpha)$ **then**

$o^{(j)} = \text{vrai}$

else

$o^{(j)} = \text{faux}$

end

end

return \mathbf{o}

Chapitre 3

Expérimentations

Nous avons testé notre approche de détection d'anomalies sur différents jeux de données. Nous avons également comparé nos résultats avec d'autres méthodes. Dans tous les cas, l'objectif est d'utiliser un jeu de données d'entraînement \mathcal{X} contenant une proportion p d'anomalies et de tester le modèle sur un jeu de données de test \mathcal{X}^* contenant une proportion p^* anomalies. Toutes les approches comparées sont des méthodes non-supervisées.

3.1 Jeux de données

Dans la première expérimentation, notre jeu de données a été créé à partir d'images réelles provenant de différentes sources de données, dont principalement *ImageNet* (Deng et al., 2009). Pour simplifier la terminologie, nous allons utiliser le terme *ImageNet* pour caractériser le jeu de données de cette expérience. Dans la deuxième expérimentation, notre jeu de données a été créé à partir d'images tirées de *MNIST* (LeCun et al., 2010). Dans les deux cas, les anomalies sont définies à partir des classes du jeu de données. Nous utiliserons les classes seulement dans le but de valider notre approche, et non dans l'entraînement du modèle.

Dans le cas du premier jeu de données, nous avons défini la classe dite "normale" à partir des images de *Stanford Cars* (Krause et al., 2013), soit des images de voitures. Les anomalies sont des images tirées aléatoirement des jeux de données *Stanford Dogs* (Khosla et al., 2011) et *Indoor scene recognition* (Quattoni and Torralba, 2009), des images de chiens et de pièces de bâtiment, respectivement. La figure 3.1 présente quelques exemples d'images provenant des 2 classes. Les images sont de type *RGB* et ont toutes été re-dimensionnées en des images de 128 pixels par 128 pixels. Le tableau 3.1 montre le nombre d'instances utilisées dans les jeux de données d'entraînement \mathcal{X} et de test \mathcal{X}^* . On peut également y voir la proportion d'anomalies dans chaque ensemble selon 3 scénario différents. Le scénario de contamination appelé "Moins", signifie qu'il y a moins d'anomalies dans l'ensemble d'entraînement que dans l'ensemble de test. Le scénario appelé "égal" veut dire qu'il y a la même proportion d'anomalies dans les deux ensembles. Finalement, le scénario appelé "Plus", signifie qu'il y a plus d'anomalies dans

l'ensemble d'entraînement que dans l'ensemble de test.

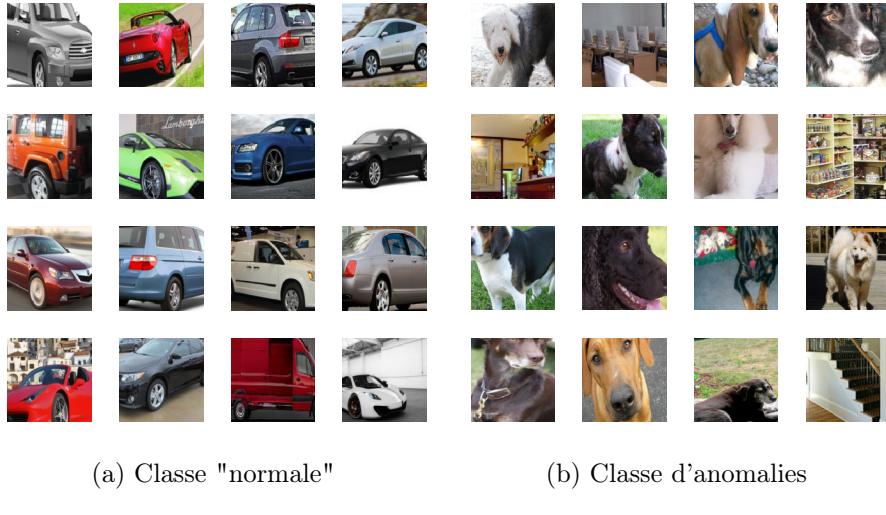


FIGURE 3.1 – Exemples d’images pour notre jeu de données d’images réelles tirées de *ImageNet*. La figure à gauche (a) correspond à des exemples de la classe dite "normale" et le volet de droite (b) correspond à des exemples de la classe d'anomalies.

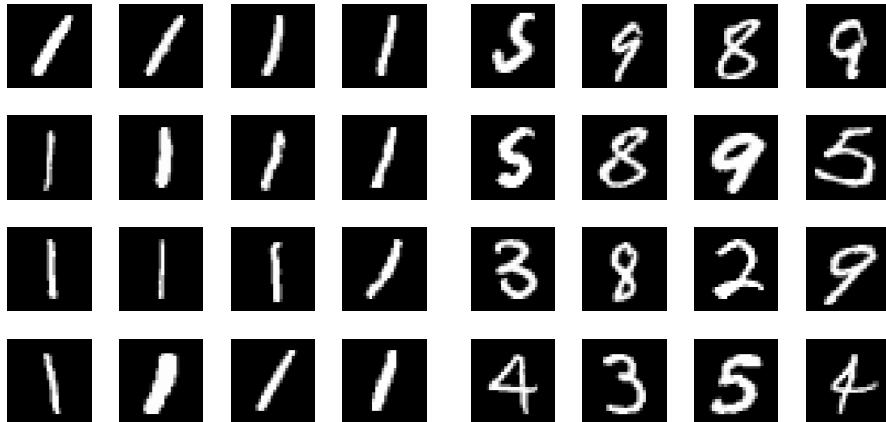
TABLE 3.1 – Description des 2 ensembles de données pour le jeu de données de *ImageNet*.

Contamination	Ensemble de données	Nombre d’instances	Pourcentage d’anomalies
Moins	\mathcal{X}	10000	10%
	\mathcal{X}^*	1000	1%
Égal	\mathcal{X}	10000	5%
	\mathcal{X}^*	1000	5%
Plus	\mathcal{X}	10000	1%
	\mathcal{X}^*	1000	10%

Dans le cas du deuxième jeu de données, la classe "normale" est définie en prenant une catégorie du jeu de données *MNIST*, soit un chiffre de 0 à 9. Nous avons ensuite défini 3 scénarios de test différents. Ces 3 scénarios se différencient par le nombre de classes que nous catégorisons comme des anomalies. Pour chacun de ces 3 scénarios, nous avons testé 2 catégories "normales" différentes, ce qui donne au total 6 scénarios de test. Ces scénarios sont définis plus en détails dans le tableau 3.2. La figure 3.2 présente quelques exemples d’images, considérant que le chiffre 0 est la classe "normale" et tous les autres chiffres sont des anomalies. Les images sont de type noir et blanc et sont toutes de dimensions 28 pixels par 28 pixels. Le tableau 3.3 montre le nombre d’instances utilisées dans les jeux de données d’entraînement \mathcal{X} et de test \mathcal{X}^* . Il montre également les proportions d’anomalies dans chaque ensemble, de la même manière qu’avec le jeu de données *ImageNet*. Pour chacun des niveaux de contamination, les 6 différents scénarios du tableau 3.2 ont été testés.

TABLE 3.2 – Description des 2 ensembles de données pour le jeu de données provenant de *MNIST*.

Scénario de test	Chiffre "normal"	Chiffres "anormaux"
1	1	5
2	1	5,9
3	1	0,2,3,4,5,6,7,8,9
4	6	8
5	6	3,8
6	6	0,1,2,3,4,5,7,8,9



(a) Classe normale

(b) Classe d'anomalies

FIGURE 3.2 – Exemples d'images pour notre jeu de données *MNIST*. La figure à gauche (a) correspond à des exemples de la classe "normale" et le volet de droite (b) correspond à des exemples de la classe d'anomalies. Il s'agit plus particulièrement du scénario de test 3.

Contamination	Ensemble de données	Nombre d'instances	Pourcentage d'anomalies
Moins	\mathcal{X}	4000	10%
	\mathcal{X}^*	800	1%
Égal	\mathcal{X}	4000	5%
	\mathcal{X}^*	800	5%
Plus	\mathcal{X}	4000	1%
	\mathcal{X}^*	800	10%

TABLE 3.3 – Description des 2 ensembles de données pour le jeu de données de *MNIST*.

3.2 Méthodes testées

Dans un premier temps, nous avons testé notre approche de détection d'anomalies basée sur des autoencodeurs variationnels, que nous appellerons DA-VAE (section 3.2.1). Dans un deuxième temps, nous avons testé trois autres approches. Tout d'abord, nous avons testé une méthode basée sur une analyse en composantes principales (ACP), décrite plus en détails dans la section 3.2.2. Ensuite, nous avons testé une approche basée sur un autoencodeur traditionnel avec des

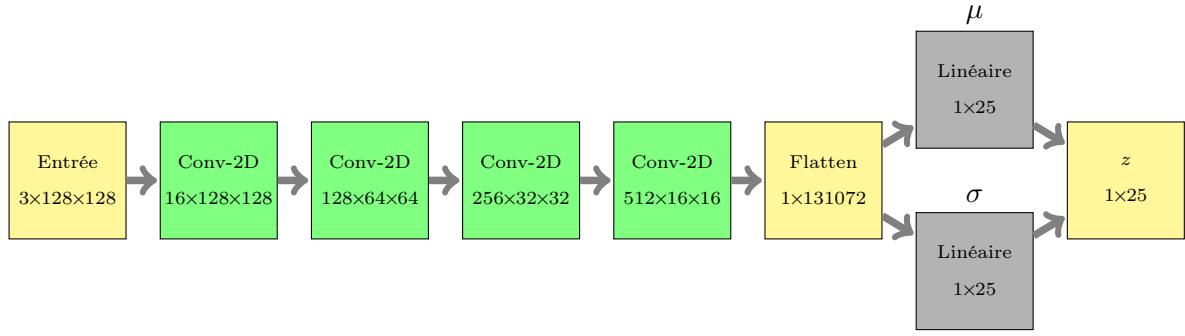
couches de convolutions (AE), décrite dans la section 3.2.3. Finalement, nous avons testé une dernière approche utilisant le même autoencodeur variationnel que dans l’approche DA-VAE, mais en appliquant l’algorithme de détection d’anomalies *Isolation Forest* (Liu et al., 2008) sur les représentations latentes (ISOF-VAE). Cette méthode est décrite plus en détails dans la section 3.2.4. Ces quatre différentes approches et leurs détails d’entraînement sont approfondis dans les prochaines sous-sections.

3.2.1 Détails sur la méthode DA-VAE

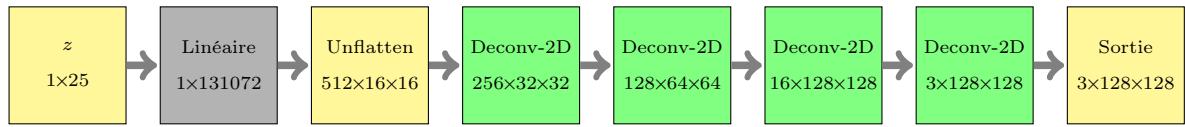
La section 2 décrit en détails le fonctionnement de l’approche que nous proposons. Par contre, certains détails concernant l’entraînement de l’autoencodeur variationnel ont été omis jusqu’à maintenant. Tout d’abord, il faut mentionner que l’architecture du réseau de neurones est différente pour chacun des 2 jeux de données décrits à la section 3.1.

Dans le cas du jeu de données provenant de *ImageNet*, nous avons défini 4 couches de convolution pour l’encodeur et 4 couches de convolution transposée pour le décodeur. Chaque couche de convolution est suivie d’une normalisation par *batch* (*batch normalization*) et d’une fonction d’activation de type *ReLU*. Juste avant de définir les couches de vecteurs latents μ et σ , le résultat de la dernière couche de convolution de l’encodeur, de dimensions $(512 \times 16 \times 16)$, est aplati en un vecteur de longueur 131 072. Les couches de vecteurs latents μ et σ sont ensuite obtenue via une couche linéaire. La couche latente possède 25 dimensions. Nous avons entraîné le réseau sur 25 itérations complètes des données (25 *epochs*) tout en ayant 130 observations par mini-batch. L’algorithme d’optimisation est celui de *Adam* (Kingma and Ba, 2014) et le taux d’apprentissage (*learning rate*) est de 0.001. Finalement, le niveau de filtration α appliqué sur le jeu de données de test est le même que la proportion de contamination dans le jeu de donnée de test. La figure 3.3 présente en détails les différentes couches de l’encodeur et du décodeur.

Pour le jeu de données provenant de *MNIST*, nous avons défini 3 couches de convolution pour l’encodeur et 3 couches de convolution transposée pour le décodeur. Chaque couche de convolution est suivie d’une fonction d’activation de type *ReLU*. Chaque couche de convolution de l’encodeur est également suivie d’un *max pooling* de dimensions 2×2 . Juste avant de définir les couches de vecteurs latents μ et σ , le résultat de la dernière couche de convolution de l’encodeur, de dimensions $(64 \times 4 \times 4)$, est aplati en un vecteur de longueur 1024. Les couches de vecteurs latents μ et σ sont ensuite obtenues via une couche linéaire. La couche latente possède 25 dimensions. Nous avons entraîné le réseau pendant 75 itérations complètes des données (75 *epochs*) tout en ayant 256 observations par mini-batch. L’algorithme d’optimisation est celui de *Adam* et le taux d’apprentissage (*learning rate*) est de 0.001. Finalement, le niveau de filtration α appliqué sur le jeu de données de test est le même que la proportion de contamination dans l’ensemble de données de test. La figure 3.4 présente en détails les différentes couches de l’encodeur et du décodeur.

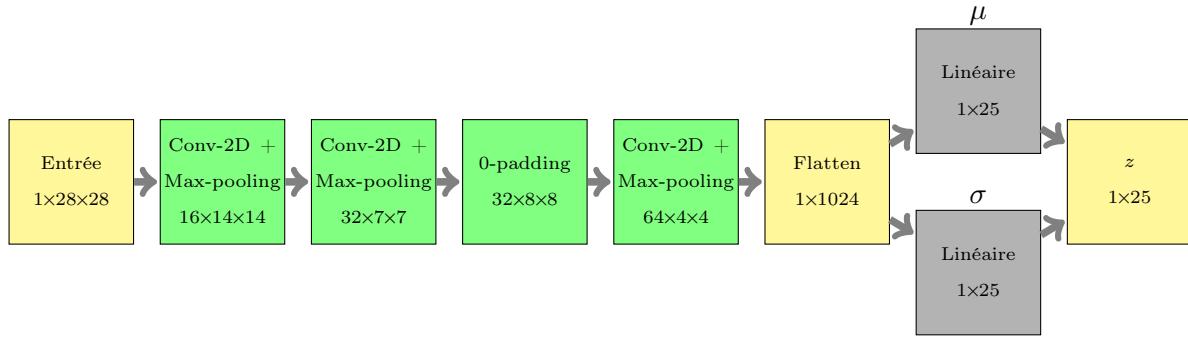


(a) Architecture de l'encodeur

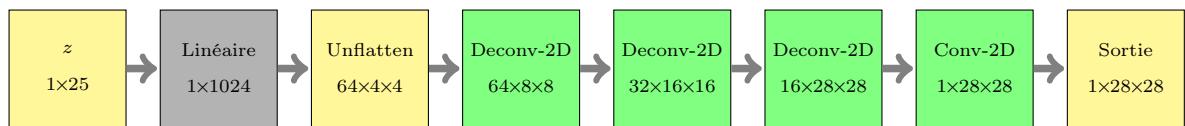


(b) Architecture du décodeur

FIGURE 3.3 – Architecture du modèle DA-VAE pour le jeu de données *ImageNet*. Les blocs représentent les couches du réseau. Les blocs jaunes représentent les couches où il n'y a pas de paramètres à optimiser. Les blocs verts sont des couches de convolution ou de déconvolution et les blocs gris sont des couches linéaires.



(a) Architecture de l'encodeur



(b) Architecture du décodeur

FIGURE 3.4 – Architecture du modèle DA-VAE pour le jeu de données provenant de *MNIST*. Les blocs représentent les couches du réseau. Les blocs jaunes représentent les couches où il n'y a pas de paramètres à optimiser. Les blocs verts sont des couches de convolution ou de déconvolution et les blocs gris sont des couches linéaires.

Horaire du paramètre β

Dans les deux expériences différentes, nous avons traité le paramètre β d'une manière particulière. En effet, étant donné que nous utilisons des β -VAE, nous devons définir un paramètre β qui donne plus ou moins d'importance à la composante de perte associée au critère de Kulbach-Leibler (voir section 1.3.1). Dans notre cas, nous n'avons pas conservé ce paramètre β fixe tout au long de l'entraînement. Pour l'entraînement des 2 différents jeux de données, nous avons donc mis en place un horaire du paramètre β selon l'itération d'entraînement (ou *epoch*). Cet horaire n'est pas le même selon le jeu de données et a été choisi en se basant sur la section 3.1 de l'article Bowman et al. (2016). Ce qui est proposé dans cet article est de commencer l'entraînement en attribuant un poids de zéro à la perte de KL. Pendant les premières itérations, le réseau va donc optimiser ses paramètres en fonction de la composante de reconstruction seulement. Par la suite, on augmente le paramètre β pour régulariser le modèle et ainsi rapprocher les paramètres μ et σ de ceux de la $N(0, I)$. Le tableau 3.4 présente les paramètres β utilisés selon l'itération (*epoch*) d'entraînement et le jeu de données.

TABLE 3.4 – Horaire des paramètres β pour le jeu de données provenant de *ImageNet* et le jeu de données provenant de *MNIST*.

Itération	$\beta_{ImageNet}$	β_{MNIST}
[0, 5]	0	0
[6, 10]	100	10
[11, ∞]	10	1

L'intuition derrière l'horaire établi est de contrôler le modèle dans les premières itérations pour éviter d'accorder trop d'importance, et ce trop rapidement, à la composante de régulation KL. Cela pourrait avoir comme conséquence de nuire au reste de l'optimisation et à l'apprentissage de la composante de reconstruction.

3.2.2 Détails sur la méthode ACP

Une première approche testée pour comparer nos résultats consiste à effectuer une analyse en composantes principales (ACP). L'objectif premier de cette technique non-supervisée est de représenter les données dans un espace plus simple. Il est aussi possible de reconstruire ces représentations dans les dimensions originales, et ainsi se calculer un score d'anomalie basé sur la capacité de bien reconstruire. Dans cette méthode, les valeurs et vecteurs propres de la matrice de covariance des variables standardisées

$$\text{var}(\mathbf{X}) = \frac{1}{n} \mathbf{X}^T \mathbf{X}, \quad (3.1)$$

nous permettent d'obtenir une projection de cette matrice dans un espace à plus petites dimensions. La matrice de vecteurs propres \mathbf{V} est de dimensions $p \times d$, où p correspond au

nombre de variables de la matrice \mathbf{X} et d correspond au nombre de composantes principales conservées dans l'espace réduit. La représentation en dimension réduite, soit \mathbf{Z} , est obtenue par le produit de $\mathbf{X}\mathbf{V}$. Finalement, la reconstruction dans l'espace original, $\hat{\mathbf{X}}$, est donnée par

$$\hat{\mathbf{X}} = \mathbf{Z}\mathbf{V}^T = \mathbf{X}\mathbf{V}\mathbf{V}^T.$$

Étant donné que nos données sont des images, ou des matrices en plusieurs dimensions, nous avons aplati chacune des entrées pour obtenir des vecteurs. Dans le cas des images couleurs provenant de *ImageNet*, nous avons d'abord aplati chacun des canaux RGB et ensuite concaténé les 3 vecteurs ensemble. Chaque image est de dimensions $3 \times 128 \times 128$, ce qui veut dire que nous obtenons un vecteur de longueur 49152. Pour les images provenant de *MNIST*, les images sont de dimensions $1 \times 28 \times 28$, ce qui nous donne des vecteurs de longueur 784. Cette transformation est nécessaire dans le cas d'une analyse en composantes principales. Ces 2 approches de vectorisation sont illustrées dans la figure 2.1.

Pour trouver les anomalies, nous avons utilisé l'erreur de reconstruction entre les données originales $(x^{*(j)})_{j=1}^k$ et leur reconstruction $(\hat{x}^{*(j)})_{j=1}^k$ basée uniquement sur un certain nombre de composantes principales. Dans nos expériences, ce nombre de composantes principales à été établi afin d'obtenir une projection conservant approximativement 80% de la variance originale des données. Nous avons utilisé l'erreur quadratique moyenne comme erreur de reconstruction. Une observation $x^{*(j)}$ est considérée comme une anomalie au niveau de filtration α si

$$l(x^{*(j)}, \hat{x}^{*(j)}) > t_{1-\alpha}(L(\mathcal{X}, \hat{\mathcal{X}})), \quad (3.2)$$

où $l(x^{*(j)}, \hat{x}^{*(j)})$ correspond à l'erreur de reconstruction pour l'observation i du jeu de données test \mathcal{X}^* et $t_{1-\alpha}(L(\mathcal{X}, \hat{\mathcal{X}}))$ correspond au percentile $1 - \alpha$ de l'ensemble des erreurs de reconstruction sur le jeu de données d'entraînement \mathcal{X} . Le niveau de filtration α choisi est le même que la proportion de contamination dans l'ensemble de données de test.

Pour le jeu de données *ImageNet*, nous avons conservé les 100 premières composantes principales. Pour le jeu de données *MNIST*, nous avons conservé les 30 premières composantes principales.

3.2.3 Détails sur la méthode AE

Une deuxième approche testée consiste à utiliser un autoencodeur traditionnel et de se servir de l'erreur de reconstruction comme score d'anomalie. Pour cette approche, nous avons utilisé les mêmes architectures que celles définies dans la section 3.2.1, mais sans les couches linéaires

latentes du réseau DA-VAE. Par exemple, pour l'autoencodeur appliqué sur les images provenant de *ImageNet*, nous avons conservé les 4 premières couches de convolution et avons ensuite appliqué les 4 dernières couches de convolution transposée immédiatement après. Il n'y a donc pas de vectorisation qui est nécessaire.

Nous avons utilisé l'erreur quadratique moyenne comme erreur de reconstruction. Les anomalies du jeu de données test \mathcal{X}^* sont donc prédites de façon similaire à ce qui est présenté à l'équation 3.2. Le niveau de filtration α choisi est le même que la proportion de contamination dans le jeu de données de test.

3.2.4 Détails sur la méthode ISOF-VAE

Finalement, nous avons testé une dernière approche qui consiste à utiliser différemment les représentations latentes du réseau entraîné avec l'approche DA-VAE. Au lieu de calculer une distance de Kullback-Leibler entre chaque représentation latente et une distribution normale $N(0, I)$, nous utiliserons ici les représentations latentes pour entraîner l'algorithme *Isolation Forest*. Cet algorithme non-supervisé, principalement utilisé en détection d'anomalie, utilise plusieurs arbres de décisions binaires regroupés dans ce qu'on appelle une forêt d'arbres. Chacun des arbres est entraîné sur un certains nombre d'observations tiré au hasard et avec remise. De plus, il est également possible de tirer au hasard un certains nombre de variables pour entraîner chacun des arbres. Ce processus est très similaire à ce qui est fait en apprentissage supervisé avec les forêts aléatoires.(Statistics and Breiman, 2001). L'hypothèse de base de cet algorithme quant aux anomalies est fondée autour du fait qu'elles sont peu nombreuses et possèdent des caractéristiques différentes des observations "normales". Sachant cette hypothèse, ces anomalies sont susceptibles d'être isolées dans les premières séparations des arbres de décisions. Pour déterminer si une observation est isolée, on calcule le nombre de noeuds à parcourir dans l'arbre pour atteindre cette observation. Plus ce chemin est petit, plus l'observation risque d'être une anomalie.

Afin d'obtenir notre propre score d'anomalie comparable à notre niveau de filtration α , nous avons légèrement transformé le score d'anomalie calculé par l'algorithme. Ce score S est calculé pour une observation en prenant la moyenne du nombre de noeuds parcouru dans chacun des arbres de la forêt pour atteindre cette observation. Plus ce score est petit, plus l'observation est susceptible d'être isolée et d'être considérée "anormale". Pour être en mesure de comparer ce score avec notre niveau de filtration, nous avons d'abord calculé ce score pour chacune des n observations de notre jeu de données d'entraînement, ce que nous appelons l'ensemble $S_{\mathcal{X}} = \{S^{(1)}, \dots, S^{(n)}\}$. Nous définissons également $S'_{\mathcal{X}}$ comme l'ensemble $S_{\mathcal{X}}$ ordonné en ordre croissant. Ensuite, nous avons défini notre propre score d'anomalie $S^*(\cdot)$ pour une observation du jeu de donnée de test $x^{*(j)}$ comme suit,

$$S^*(x^{*(j)}) = \frac{rang_{S'_\chi}(S^{(j)})}{n}, \quad (3.3)$$

où $rang_{S'_\chi}(S^{(j)})$ correspond au rang du score $S^{(j)}$ de l'observation $x^{*(j)}$ dans l'ensemble ordonné S'_χ . Ce score $S^*(x^{*(j)})$, peut être comparé à notre niveau de filtration α . Ce niveau de filtration choisi est le même que la proportion de contamination dans le jeu de données de test.

3.3 Résultats

Pour comparer les différentes méthodes, nous avons utilisé 3 métriques différentes. La première est l'aire sous la courbe ROC. Cette métrique nous permet de faire abstraction du seuil de filtration α . Ensuite, nous avons également choisi les métriques de précision et de rappel, qui prennent en considération un certain seuil, α dans notre cas. Nous avons entraîné et testé les algorithmes à 20 reprises, ce qui nous permet de calculer une moyenne et une mesure de dispersion, soit l'écart-type, autour de chacune des métriques. Pour les 20 expérimentations, l'échantillonnage des jeux de données de test et d'entraînement est refait. La variabilité des résultats entre les 20 expérimentations est expliquée par 3 composantes :

1. L'échantillonnage des jeux de données ;
2. L'initialisation des paramètres et l'optimisation par descente du gradient dans les réseaux de neurones ;
3. La composante stochastique dans la représentation latente du VAE.

La première composante, soit l'échantillonnage des jeux de données, affecte les 4 différents algorithmes. La deuxième composante, soit l'initialisation des paramètres et l'optimisation par descente du gradient, n'affecte que les méthodes reliées à des réseaux de neurones : DA-VAE (3.2.1), AE (3.2.3) et ISOF-VAE (3.2.4). La composante stochastique dans l'optimisation par descente du gradient est reliée au fait que le jeu de données d'entraînement est divisé en sous-ensembles, aussi appelés *mini-batch*, qui sont définis aléatoirement. Ces sous-ensembles sont utilisés pour calculer les pertes sur lesquelles on calcule les dérivés permettant de mettre à jour les paramètres de l'autoencodeur à chaque itération d'entraînement. L'optimisation globale du réseau de neurones peut donc emprunter des chemins légèrement différents avec une même initialisation. Finalement, la troisième composante concerne seulement les deux méthodes utilisant l'autoencodeur variationnel : DA-VAE et ISOF-VAE. Comme il a été expliqué à la section 1.3, la représentation latente de l'autoencodeur variationnel est obtenue en combinant les paramètres des couches μ et σ du modèle ainsi que la simulation d'une loi $N(0, I)$.

Nous ne faisons qu'une présentation brute des résultats dans cette sous-section et nous en faisons l'analyse et la discussion à la sous-section 3.4.

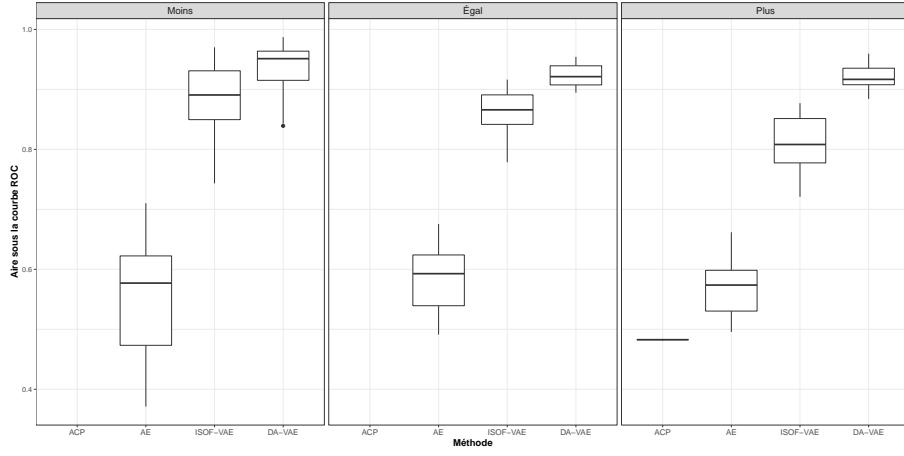
3.3.1 Résultats sur *ImageNet*

Le tableau 3.5 synthétise les résultats obtenus sur le jeu de données provenant de *ImageNet*. On peut y voir les résultats pour les 4 approches différentes, pour les 3 scénarios de contamination ainsi que pour les 3 métriques choisies. Nous avons mis en gras les résultats de la méthode nous apparaissant la plus adaptée pour chaque métrique. Ce choix est basé en grande partie sur la valeur moyenne de la métrique.

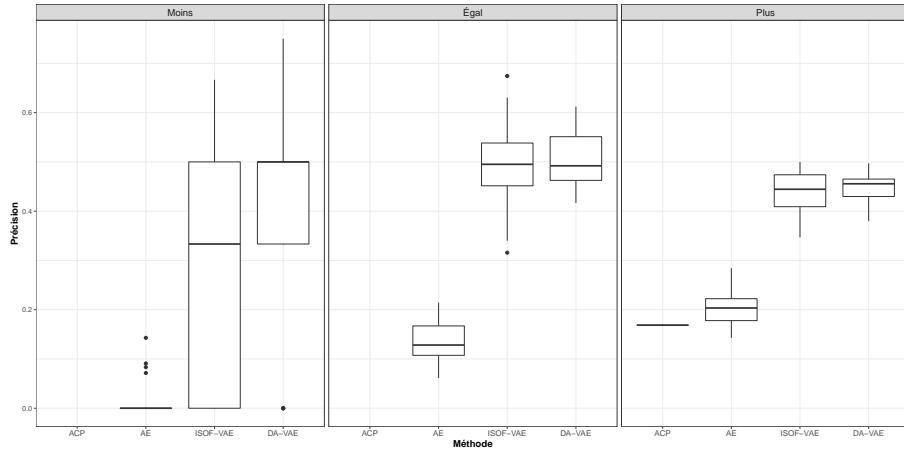
TABLE 3.5 – Résultats selon les métriques d'aire sous la courbe ROC, de précision et de rappel pour les différentes expérimentations concernant le jeu de données *ImageNet*.

Contamination	Métrique	ACP*	AE	ISOF-VAE	DA-VAE
Moins	ROC	0.582 ± 0	0.549 ± 0.090	0.886 ± 0.059	0.935 ± 0.042
	Précision $_{\alpha=0.01}$	0.089 ± 0	0.019 ± 0.041	0.325 ± 0.268	0.400 ± 0.231
	Rappel $_{\alpha=0.01}$	0.089 ± 0	0.020 ± 0.040	0.089 ± 0.079	0.120 ± 0.093
Égal	ROC	0.582 ± 0	0.584 ± 0.050	0.859 ± 0.037	0.923 ± 0.018
	Précision $_{\alpha=0.05}$	0.089 ± 0	0.136 ± 0.041	0.499 ± 0.086	0.504 ± 0.057
	Rappel $_{\alpha=0.05}$	0.089 ± 0	0.144 ± 0.048	0.461 ± 0.087	0.576 ± 0.065
Plus	ROC	0.582 ± 0	0.568 ± 0.047	0.811 ± 0.045	0.920 ± 0.019
	Précision $_{\alpha=0.1}$	0.089 ± 0	0.202 ± 0.035	0.437 ± 0.044	0.449 ± 0.030
	Rappel $_{\alpha=0.1}$	0.089 ± 0	0.231 ± 0.042	0.594 ± 0.078	0.798 ± 0.058

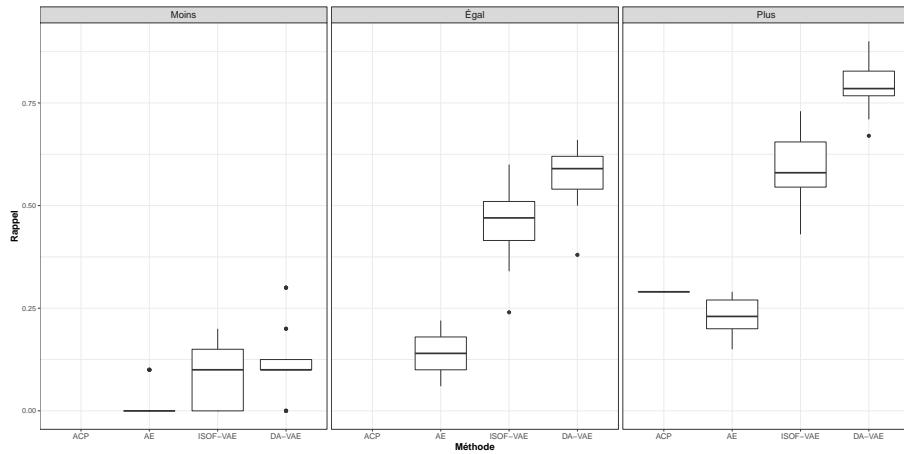
Dans la figure 3.5, on peut voir plus en détails la distribution des résultats obtenus sur les 20 expérimentations via des graphiques en boîtes et moustaches.



(a) Résultats en aire sous la courbe ROC



(b) Résultats en précision



(c) Résultats en rappel

FIGURE 3.5 – Graphiques en boîtes et moustaches illustrant les résultats sur les 20 expérimentations de chacune des approches et scénarios de contamination. Les 3 sous-figures concernent une métrique différente.

3.3.2 MNIST

Le tableau 3.6 synthétise les résultats obtenus sur le jeu de données *MNIST* pour la métrique d'aire sous la courbe ROC. Les résultats sont présentés par scénario de contamination et par scénario de test (voir tableau 3.2 pour plus de détails sur les scénarios de test). Les résultats pour les métriques de précision et de rappel sont présentés de la même manière dans les tableaux 3.7 et 3.8 respectivement. Nous avons mis en gras les résultats de la méthode nous apparaissant la plus adaptée pour chacun des scénarios. Encore une fois, ce choix est basé en grande partie sur la valeur moyenne de la métrique.

TABLE 3.6 – Résultats des aires sous la courbe ROC des différentes approches selon le scénario de contamination et le scénario de test appliquée sur le jeu de données *MNIST*.

Contamination	Scénario	ACP	AE	ISOF-VAE	DA-VAE
Moins	1	0.991 ± 0.006	0.990 ± 0.004	0.969 ± 0.031	0.987 ± 0.007
	2	0.988 ± 0.005	0.988 ± 0.005	0.965 ± 0.022	0.982 ± 0.021
	3	0.993 ± 0.005	0.994 ± 0.004	0.976 ± 0.017	0.992 ± 0.005
	4	0.893 ± 0.034	0.929 ± 0.034	0.754 ± 0.078	0.836 ± 0.067
	5	0.893 ± 0.035	0.948 ± 0.026	0.794 ± 0.061	0.880 ± 0.059
	6	0.856 ± 0.059	0.847 ± 0.092	0.850 ± 0.065	0.893 ± 0.046
Égal	1	0.993 ± 0.004	0.994 ± 0.002	0.980 ± 0.011	0.994 ± 0.002
	2	0.991 ± 0.003	0.992 ± 0.002	0.978 ± 0.010	0.988 ± 0.007
	3	0.993 ± 0.004	0.996 ± 0.001	0.980 ± 0.011	0.991 ± 0.007
	4	0.903 ± 0.027	0.959 ± 0.012	0.828 ± 0.060	0.901 ± 0.042
	5	0.913 ± 0.027	0.969 ± 0.010	0.824 ± 0.057	0.871 ± 0.122
	6	0.900 ± 0.031	0.854 ± 0.049	0.860 ± 0.056	0.921 ± 0.036
Plus	1	0.994 ± 0.003	0.998 ± 0.001	0.986 ± 0.009	0.994 ± 0.003
	2	0.993 ± 0.004	0.997 ± 0.001	0.981 ± 0.009	0.991 ± 0.007
	3	0.993 ± 0.003	0.998 ± 0.001	0.985 ± 0.008	0.995 ± 0.002
	4	0.942 ± 0.025	0.985 ± 0.005	0.833 ± 0.058	0.915 ± 0.041
	5	0.948 ± 0.021	0.992 ± 0.003	0.875 ± 0.040	0.936 ± 0.028
	6	0.929 ± 0.015	0.911 ± 0.027	0.882 ± 0.031	0.933 ± 0.028

Dans les figures 3.6, 3.7 et 3.8, on peut voir plus en détails la distribution des résultats obtenus pour les 3 différentes métriques sur les 20 expérimentations grâce à des graphiques en boîtes et moustaches.

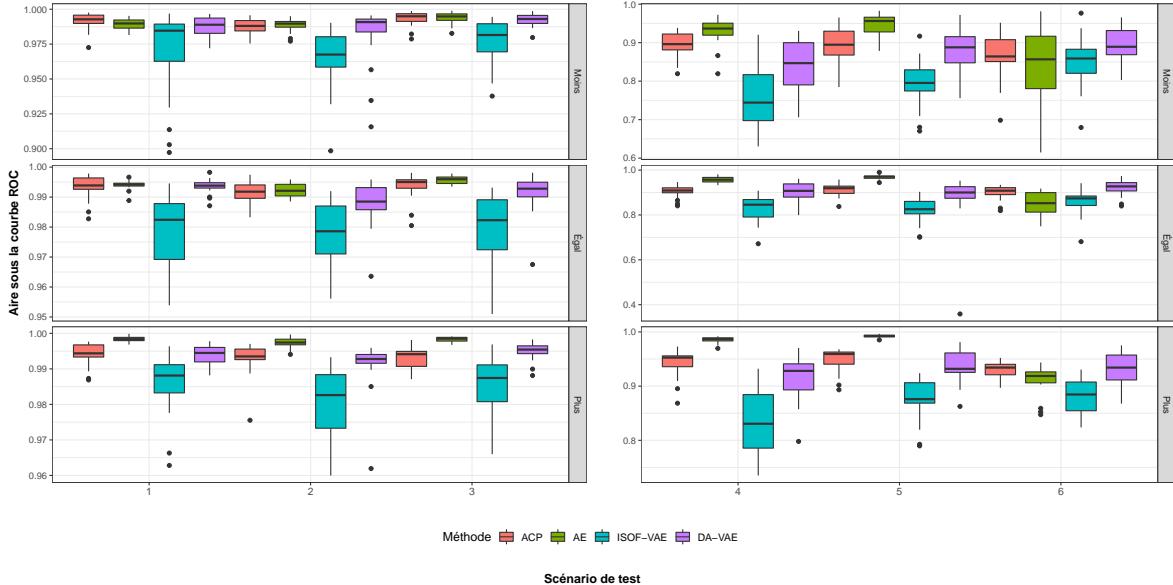


FIGURE 3.6 – Graphiques en boîtes et moustaches illustrant les résultats sur les 20 expérimentations de chacune des approches, des scénarios de test et des scénarios de contamination. Toutes les figures présentent les résultats en aire sous la courbe ROC.

TABLE 3.7 – Résultats des précisions selon la quantité de contamination et le scénario de test sur le jeu de données *MNIST*. La valeur de α dépend du scénario de contamination. Pour le scénario "Moins" $\alpha = 0.01$, pour le scénario "Égal" $\alpha = 0.05$ et finalement pour le scénario "Plus" $\alpha = 0.1$.

Contamination	Scénario	ACP	AE	ISOF-VAE	DA-VAE
Moins	1	0.688 ± 0.384	0.256 ± 0.097	0.280 ± 0.255	0.271 ± 0.212
	2	0.500 ± 0.424	0.141 ± 0.178	0.238 ± 0.234	0.282 ± 0.243
	3	0.583 ± 0.482	0.411 ± 0.319	0.263 ± 0.289	0.387 ± 0.273
	4	0.135 ± 0.182	0.197 ± 0.193	0.033 ± 0.057	0.020 ± 0.051
	5	0.110 ± 0.188	0.168 ± 0.212	0.028 ± 0.053	0.145 ± 0.159
	6	0.118 ± 0.145	0.363 ± 0.267	0.060 ± 0.092	0.115 ± 0.142
Égal	1	0.779 ± 0.079	0.777 ± 0.037	0.678 ± 0.102	0.809 ± 0.057
	2	0.685 ± 0.013	0.672 ± 0.014	0.677 ± 0.011	0.681 ± 0.009
	3	0.821 ± 0.056	0.793 ± 0.033	0.702 ± 0.089	0.829 ± 0.059
	4	0.276 ± 0.078	0.526 ± 0.063	0.216 ± 0.096	0.375 ± 0.108
	5	0.329 ± 0.081	0.556 ± 0.050	0.210 ± 0.097	0.358 ± 0.104
	6	0.331 ± 0.065	0.501 ± 0.056	0.283 ± 0.094	0.443 ± 0.113
Plus	1	0.535 ± 0.016	0.563 ± 0.020	0.545 ± 0.021	0.555 ± 0.027
	2	0.583 ± 0.024	0.566 ± 0.021	0.550 ± 0.036	0.556 ± 0.025
	3	0.531 ± 0.025	0.571 ± 0.016	0.540 ± 0.025	0.559 ± 0.018
	4	0.455 ± 0.048	0.496 ± 0.020	0.321 ± 0.083	0.424 ± 0.057
	5	0.452 ± 0.039	0.507 ± 0.020	0.379 ± 0.061	0.452 ± 0.050
	6	0.425 ± 0.023	0.464 ± 0.022	0.394 ± 0.044	0.456 ± 0.036

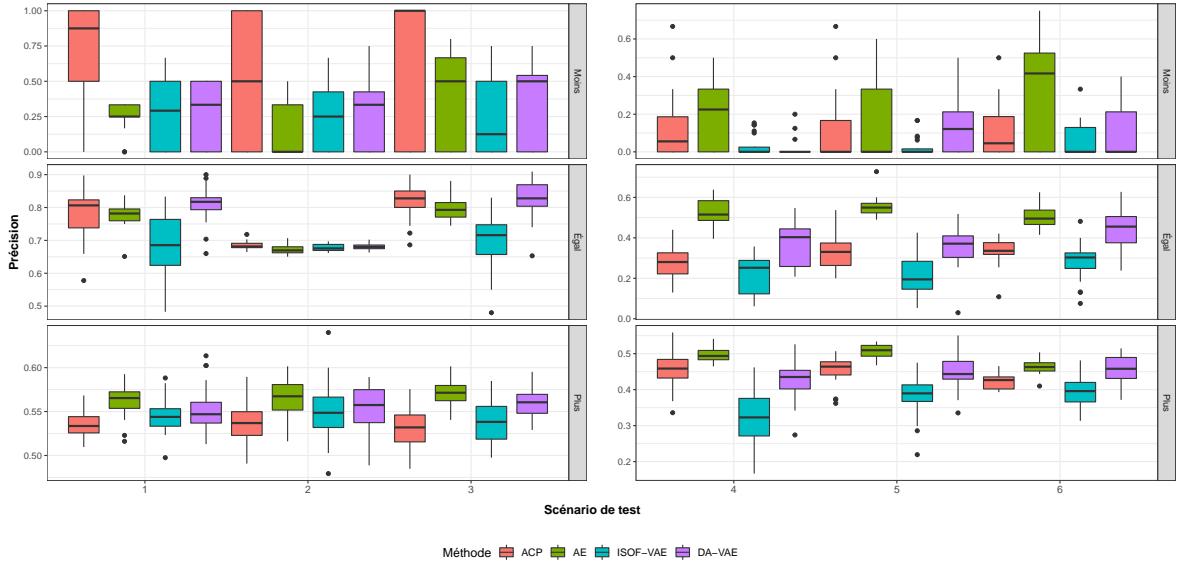


FIGURE 3.7 – Graphiques en boîtes et moustaches illustrant les résultats sur les 20 expérimentations de chacune des approches, des scénarios de test et des scénarios de contamination. Toutes les figures présentent les résultats en précision.

TABLE 3.8 – Résultats des rappels selon la quantité de contamination et le scénario de test sur le jeu de données *MNIST*. La valeur de α dépend du scénario de contamination. Pour le scénario "Moins" $\alpha = 0.01$, pour le scénario "Égal" $\alpha = 0.05$ et finalement pour le scénario "Plus" $\alpha = 0.1$.

Contamination	Scénario	ACP	AE	ISOF-VAE	DA-VAE
Moins	1	0.200 ± 0.139	0.138 ± 0.067	0.095 ± 0.092	0.075 ± 0.062
	2	0.100 ± 0.085	0.081 ± 0.107	0.085 ± 0.079	0.1 ± 0.095
	3	0.131 ± 0.134	0.181 ± 0.170	0.09 ± 0.118	0.11 ± 0.094
	4	0.100 ± 0.116	0.088 ± 0.089	0.038 ± 0.070	0.025 ± 0.063
	5	0.063 ± 0.093	0.075 ± 0.108	0.031 ± 0.054	0.088 ± 0.089
	6	0.081 ± 0.099	0.156 ± 0.130	0.056 ± 0.084	0.0625 ± 0.074
Égal	1	0.859 ± 0.083	0.871 ± 0.067	0.663 ± 0.110	0.778 ± 0.066
	2	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.001	1.000 ± 0.001
	3	0.873 ± 0.079	0.901 ± 0.038	0.671 ± 0.116	0.786 ± 0.064
	4	0.381 ± 0.096	0.628 ± 0.090	0.239 ± 0.110	0.412 ± 0.123
	5	0.451 ± 0.123	0.645 ± 0.079	0.230 ± 0.099	0.401 ± 0.112
	6	0.435 ± 0.100	0.588 ± 0.076	0.319 ± 0.101	0.481 ± 0.120
Plus	1	0.997 ± 0.005	1.000 ± 0.000	0.980 ± 0.028	0.995 ± 0.007
	2	0.994 ± 0.011	1.000 ± 0.000	0.963 ± 0.031	0.985 ± 0.027
	3	0.994 ± 0.008	1.000 ± 0.000	0.977 ± 0.023	0.996 ± 0.010
	4	0.841 ± 0.085	0.984 ± 0.012	0.476 ± 0.159	0.733 ± 0.137
	5	0.873 ± 0.091	0.994 ± 0.007	0.593 ± 0.139	0.824 ± 0.102
	6	0.789 ± 0.055	0.859 ± 0.047	0.629 ± 0.104	0.807 ± 0.097

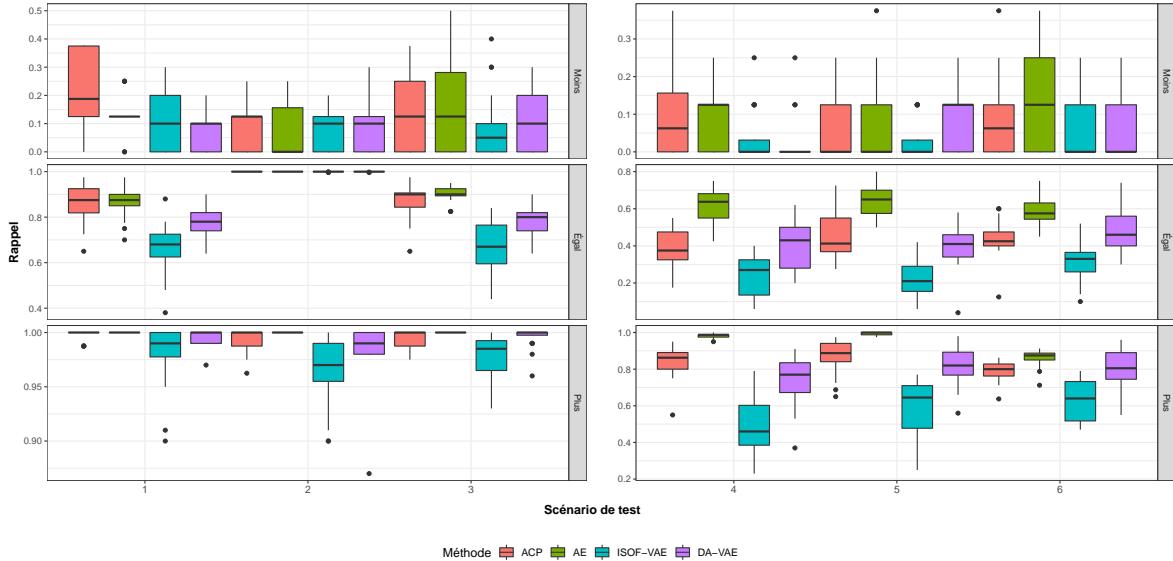


FIGURE 3.8 – Graphiques en boîtes et moustaches illustrant les résultats sur les 20 expérimentations de chacune des approches, des scénarios de test et des scénarios de contamination. Toutes les figures présentent les résultats en rappel.

3.4 Discussion

Dans les prochaines sous-sections, nous discuterons des résultats obtenus par les différentes méthodes sur les deux jeux de données. Nous irons également plus en profondeur dans l’analyse de la méthode DA-VAE en regardant le comportement de la perte en entraînement ainsi que la composition des représentations latentes produites par le modèle. Finalement, nous étudierons l’impact du niveau de contamination sur les résultats et la pertinence du niveau de filtration α .

3.4.1 Résultats sur *ImageNet*

Dans le tableau 3.5, on peut voir les performances en aire sous la courbe ROC, en précision et en rappel de chacune des approches. Pour les 3 scénarios de contamination différents, on remarque que 2 méthodes sont beaucoup plus performantes, et ce, pour les 3 métriques. Ces 2 approches sont : ISOF-VAE et DA-VAE. Il est intéressant de remarquer que ces 2 approches utilisent la représentation latente de l’image pour discriminer les anomalies des observations "normales" plutôt que la reconstruction faite sur les dimensions originales de l’image. Dans le cas spécifique avec les données de *ImageNet*, les images sont de dimensions $128 \times 128 \times 3$, ce qui veut dire que la discrimination des méthodes ACP et AE est basée sur un vecteur de longueur 49 152. À l’inverse, les méthodes ISOF-VAE et DA-VAE utilisent plutôt les vecteurs μ et σ , chacun de longueur 25.

Notre approche DA-VAE est celle qui est la plus performante selon les métriques moyennes

analysées. En tenant compte de la variabilité, il n'est cependant pas toujours possible d'affirmer que la méthode DA-VAE est significativement supérieure à la méthode ISOF-VAE. Si on ajoute 2 écarts-types au métriques moyennes obtenues par ISOF-VAE, cette borne supérieure chevauche souvent la borne inférieure de la méthode DA-VAE. On peut toutefois remarquer que les performances en aire sous la courbe ROC de la méthode DA-VAE demeurent élevées, et ce, pour les 3 scénarios de contamination. Dans le cas de ISOF-VAE, les performances moyennes en aire sous la courbe ROC passent de 0.886 pour le scénario "Moins" à 0.811 pour le scénario "Plus". La variabilité des résultats de notre approche DA-VAE est également inférieure à ceux obtenus sur ISOF-VAE pour les trois métriques. En analysant davantage les résultats sur les trois différents scénarios de contamination, on peut remarquer que pour toutes les approches, le scénario de contamination "Moins" est celui qui témoigne de la plus grande volatilité dans ses performances en test. Cela était prévisible sachant qu'il n'y a que très peu d'anomalies, soit 1%, dans le jeu de données de test. Dans les 3 sous-figures de la figure 3.9, on peut voir les scores d'anomalie prédits pour chacune des observations du jeu de données test en fonction de si l'observation est "anormale" ou non, et ce, pour les 3 scénarios de contamination. Chacun des points correspond à une observation. Les observations sont ordonnées en ordre croissant selon l'inverse de leur score d'anomalie, soit $1 - \gamma$. Ce score d'anomalie est défini à l'équation 2.10. Entre d'autres mots, plus une observation est située à gauche sur l'axe des x , plus elle est "anormale" selon le modèle DA-VAE. Les points de couleur violette sont des observations que nous connaissons comme "normales" alors que les points jaunes sont des observations que nous connaissons comme "anormales". Le trait horizontal rouge correspond à la valeur de notre niveau de filtration α . Avec ce niveau de filtration, les observations sous la droite rouge sont considérées "anormales" alors que celles au-dessus sont considérées comme "normales" selon le modèle. La ligne verte correspond simplement à un trait partant de l'origine $(0, 0)$ et allant jusqu'au point $(1, 1)$. Cette ligne nous servira de point de référence pour comparer différents scénarios de contamination.

Dans les 3 sous-figures à la figure 3.9, on peut voir que les observations "anormales", soit les points jaunes, sont principalement regroupées à gauche de l'axe des x , ce qui est le comportement souhaité. De plus, on peut remarquer que dans les 3 différents scénarios, les points les plus gauches ne se retrouvent pas à la même position par rapport à la droite verte. Dans le scénario "Égal", l'ensemble des points est relativement bien aligné avec la courbe verte. Cette observation peut s'expliquer par le fait que les deux ensembles de données, \mathcal{X} et \mathcal{X}^* , possèdent la même composition en termes de proportion d'anomalies. Dans le scénario "Moins", on peut observer une masse plus importante de points au-dessus de la courbe verte dans les premières observations. Cela est dû au fait que l'ensemble de test possède moins d'anomalies que l'ensemble d'entraînement, ce qui fait en sorte que davantage d'observations obtiennent des scores d'anomalie plus faibles. Finalement, dans le scénario "Plus", on observe qu'une masse de points se situent sous la courbe verte dans les premières observations. À l'inverse du scénario "Moins", l'ensemble de test possède ici davantage d'anomalies, en proportion, que

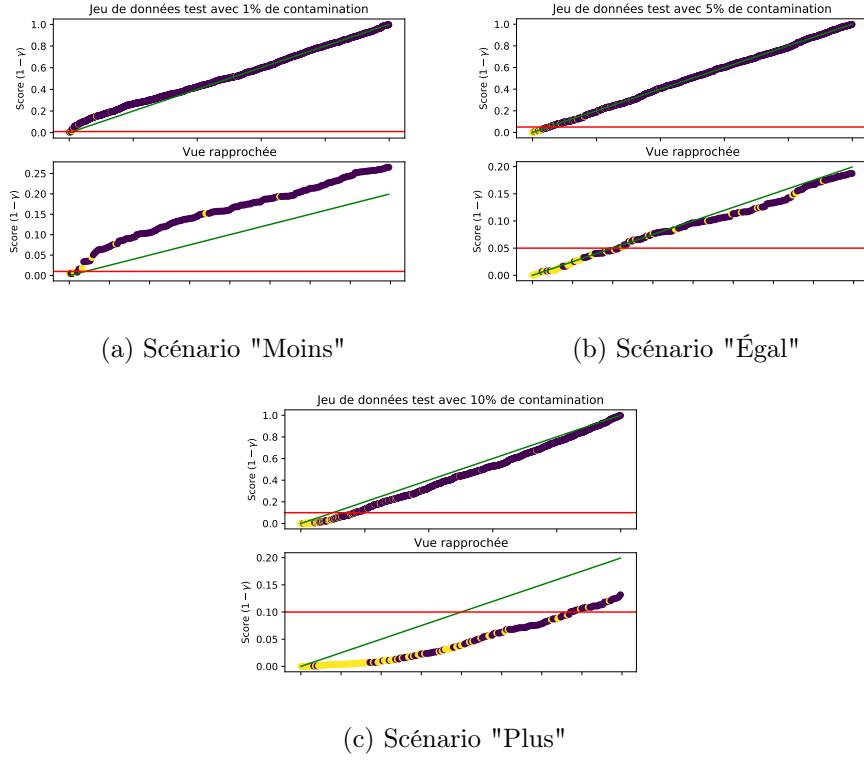


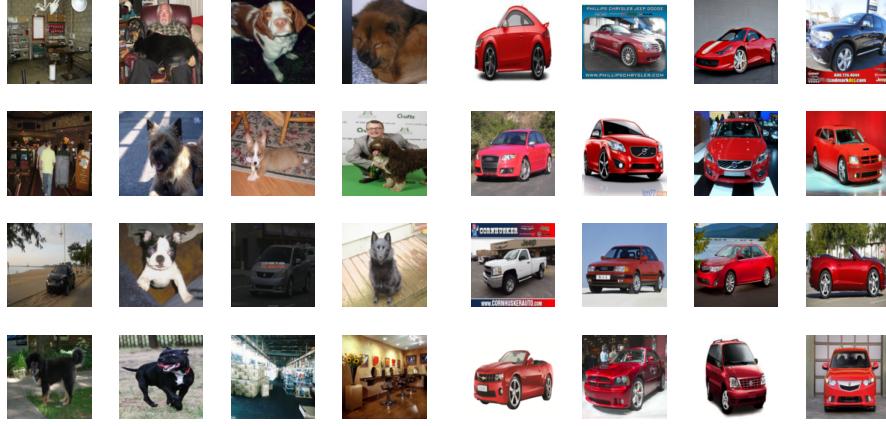
FIGURE 3.9 – Graphiques illustrant les scores d'anomalies selon le scénario de contamination pour le modèle DA-VAE appliqué sur le jeu de données *ImageNet*. Les points violet sont des observations que nous connaissons comme "normales" alors que les points jaunes sont des observations que nous connaissons comme "anormales". Dans tous les cas, le niveau de filtration α est défini comme le niveau de contamination dans le jeu de données de test.

l'ensemble d'entraînement, ce qui fait en sorte que davantage d'observations obtiennent des scores d'anomalies élevées. Cette analyse nous permet de valider visuellement que l'approche s'adapte bien à différentes proportions de contamination dans les jeux de données de test et d'entraînement.

Analyse des représentations latentes

Dans la section 2.3.2, nous avons mentionné qu'il fallait analyser les statistiques de distance afin de savoir si une statistique de distance élevée est signe d'anomalie ou de normalité. Pour se faire, il suffit de visualiser un échantillon d'images de notre jeu de données d'entraînement ayant des statistiques de distance élevées et un autre échantillon d'images ayant des statistiques de distance faibles. Dans la figure 3.10, on peut voir quelques exemples provenant de ces échantillons. On remarque assez rapidement que les observations ayant des statistiques de distance faibles proviennent de notre classe "anormale". À l'inverse, les observations ayant des statistiques de distance élevées proviennent de notre classe "normale", soit des images de voitures. On remarque que les voitures de la figure 3.10 sont presque toutes des voitures rouges.

Il est possible que les représentations latentes ce type de voitures soient situées dans un espace similaire et le plus éloigné de la $N(0, 1)$. Nous avons voulu valider si la couleur de la voiture avait un impact significatif dans la prise de décision. Pour se faire, nous avons pré-sélectionné un échantillon d'images de voitures de différentes couleurs et avons calculé le score d'anomalie pour ces images. À la figure 3.11, on peut voir que les voitures rouges obtiennent des scores d'anomalie plutôt élevé, même une avec un score sous la valeur du seuil de filtration α . Cela nous permet de confirmer que ce n'est pas la couleur de la voiture seulement qui a un impact.



(a) Statistiques de distance faible

(b) Statistiques de distance élevées

FIGURE 3.10 – Échantillons d'images provenant du jeu de données d'entraînement ayant des statistiques de distance faibles (a) et élevées (b) pour le scénario "Plus" du jeu de données *ImageNet*.

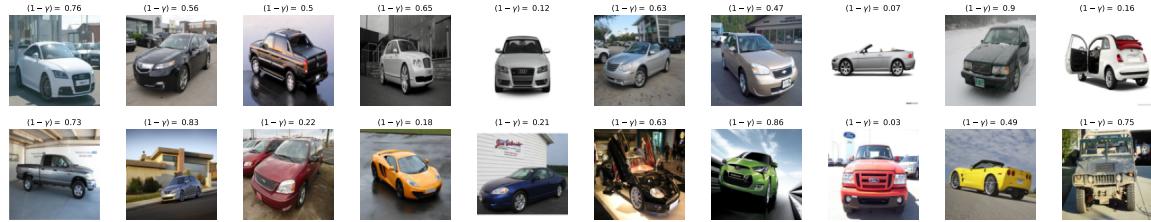


FIGURE 3.11 – Échantillons d'images pré-sélectionnées avec l'inverse de leur score d'anomalie $(1 - \gamma)$. Le score a été calculé selon le modèle DA-VAE du scénario "Plus" sur le jeu de données *ImageNet*.

De cette manière, il est possible de conclure que nous nous retrouvons dans le scénario où les représentations latentes des observations "normales" sont plus éloignées de la $N(0, I)$ (voir les deux scénarios décrits à 2.3.2). À la figure 3.12, on peut confirmer cette observation alors qu'on voit que les valeurs moyennes des μ et σ des observations "normales" sont plus éloignées de la $N(0, I)$. Dans cette figure, on calcule pour chaque observation du jeu de données d'entraînement, la valeur moyenne sur les 25 dimensions latentes des vecteurs μ et σ . La différence est beaucoup plus évidente pour le vecteur latent σ que pour μ . Pour μ , les observations

"normales" sont légèrement plus centrées à 0, mais les observations "anormales" s'éloignent moins de la valeur centrée de 0. À la figure 3.13, nous avons réalisé une analyse en composantes principales sur les vecteurs μ et σ et avons conservé les 2 premières composantes principales. Cela nous permet d'obtenir une visualisation en 2 dimensions des représentations latentes des observations "normales" et des anomalies. On peut voir que les 2 groupes sont relativement bien séparés sur les 2 premières composantes principales seulement. On peut également voir la projection en 2 dimensions du point $(\mathbf{0}_m, \mathbf{1}_m)$, soit le carré noir. On remarque que les anomalies (points bleus) semblent légèrement plus près du carré noir que les observations "normales". Il faut rappeler que la projection en 2 dimensions conserve approximativement 50% de la variabilité totale des représentations latentes.

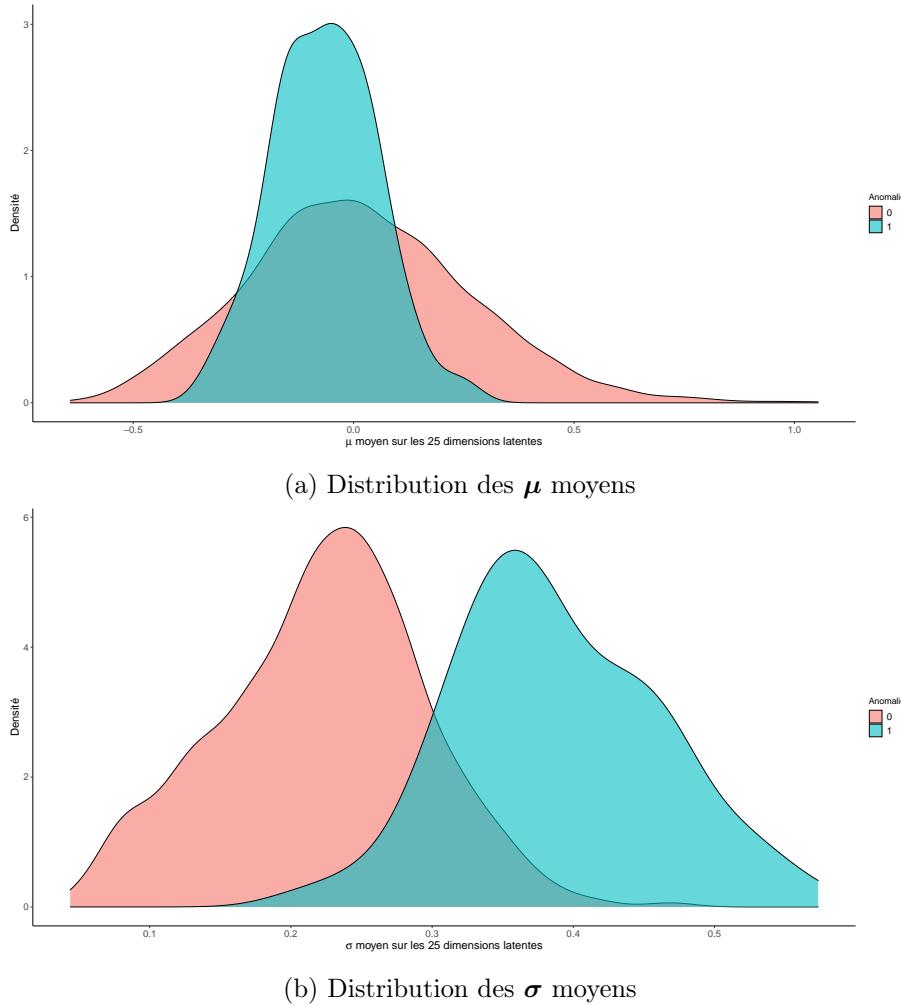


FIGURE 3.12 – Moyenne des μ et σ des représentations latentes du jeu de données d'entraînement sur le jeu de données *ImageNet*.

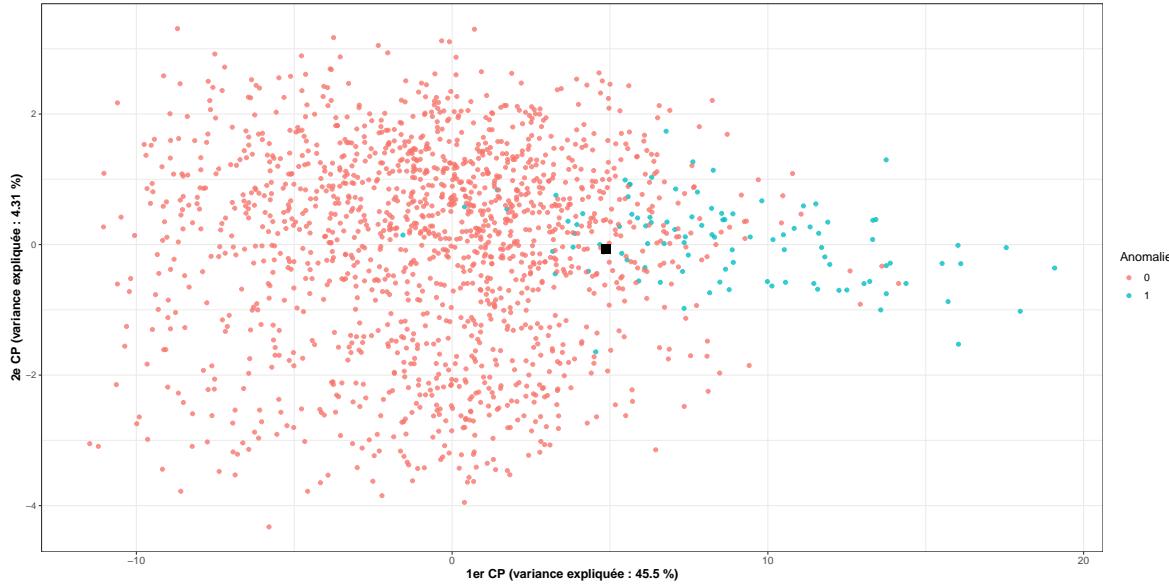


FIGURE 3.13 – Graphique des 2 premières composantes principales réalisé sur les vecteurs μ et σ du jeu de données d’entraînement de *ImageNet* appliqué sur le scénario de contamination "Plus".

Analyse de la perte en entraînement

Dans la section 3.2.1, nous avons présenté l’horaire de l’hyperparamètre β utilisé selon l’itération d’entraînement de l’autoencodeur variationnel. Cet hyperparamètre, utilisé dans les méthodes ISOF-VAE et DA-VAE, donne plus ou moins d’importance à la composante de perte associée au critère de Kullback-Leibler. Notre objectif était de mettre l’accent sur la composante de reconstruction ($\beta = 0$) dans les premières itérations pour ensuite augmenter la régularisation en augmentant l’hyperparamètre β pendant quelques itérations. La figure 3.14 illustre la composition de notre perte totale au fil des itérations d’entraînement. La figure illustre le pourcentage de la perte provenant du critère de Kullback-Leibler. L’autre composante de la perte est associée à la reconstruction. On peut voir que les 5 premières itérations sont composées uniquement de la composante de reconstruction. Cela est dû à notre hyperparamètre $\beta = 0$. Par la suite, la composante de Kullback-Leibler prend plus d’importance et tend finalement vers une valeur près de 0 dans les dernières itérations.

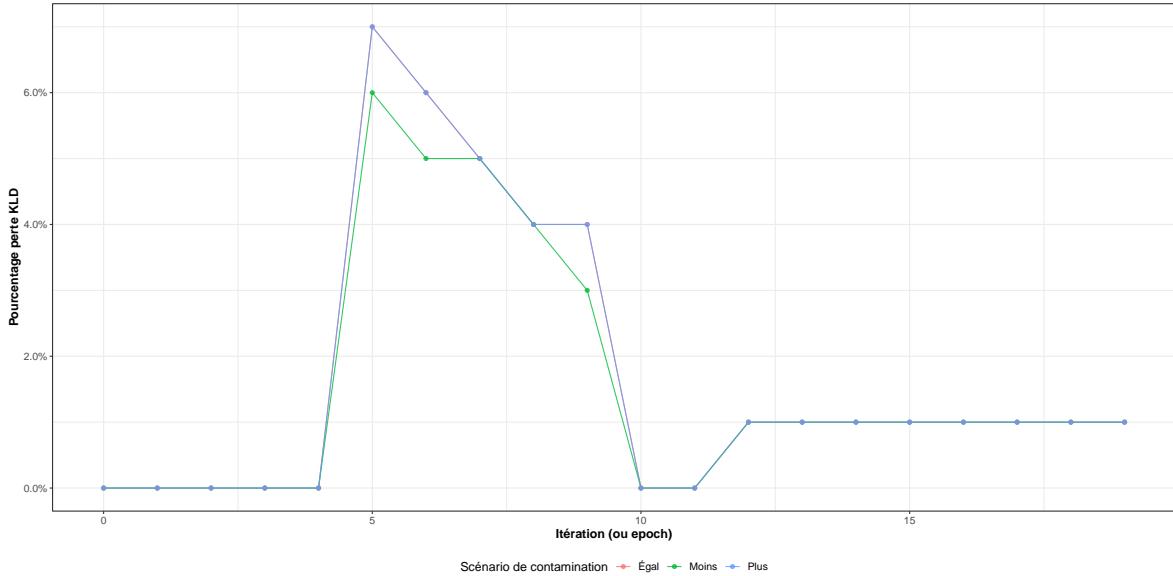


FIGURE 3.14 – Pourcentage du critère de Kullback-Leibler dans la perte totale selon l’itération d’entraînement et le scénario de contamination. Ces résultats sont tirés du scénario de contamination "Plus" pour le jeu de données *ImageNet*.

Génération d’images

Comme mentionné dans la section 1.3, les autoencodeurs variationnels ont la particularité d’avoir une représentation latente continue. En effet, cela est dû au fait que le vecteur latent est simulé à partir d’une loi $N(0, I)$ et ensuite combiné aux vecteurs μ et σ appris par le modèle. Étant donné la composante de perte de Kullback-Leibler appliquée dans l’optimisation des paramètres, ces 2 vecteurs devraient s’approcher des vecteurs $(\mathbf{0}_m, \mathbf{1}_m)$ où m correspond à la longueur de la représentation latente. Après l’entraînement de l’autoencodeur, il est possible de valider le comportement du décodeur par rapport à une simulation $N(0, I)$. La figure 3.15 montre des exemples d’images générées à partir de simulations de loi $N(0, I)$ et reconstruites par la partie décodeur du DA-VAE. Ces images, générées de toute pièce, illustrent bien qu’une représentation latente provenant d’une loi $N(0, I)$ donne un résultat qui s’apparente à une voiture ou aux caractéristiques fréquentes d’une voiture (roues, phares, pare-brise).

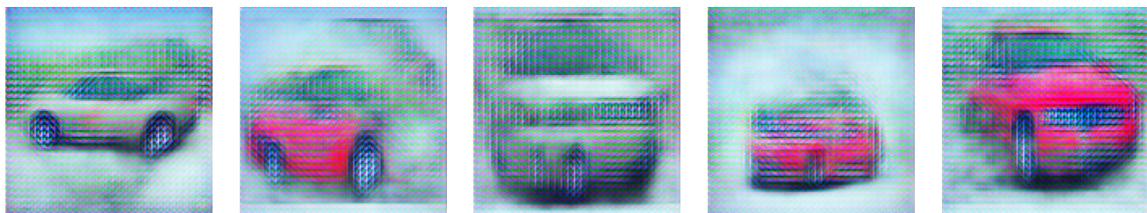


FIGURE 3.15 – Exemples d’images générées par le modèle DA-VAE en utilisant seulement la partie décodeur de l’autoencodeur ainsi que des simulations provenant d’une loi $N(0, I)$.

3.4.2 Résultats sur *MNIST*

Dans le tableau 3.6, on peut voir les performances en aire sous la courbe ROC pour les différents scénarios de test réalisés sur *MNIST*. Premièrement, on remarque que toutes les approches donnent des résultats performants avec des aires sous la courbe généralement supérieures à 0.85, et même parfois très près de 1. Cette observation vient témoigner du fait qu'étant donné la nature plus simple des données, il est probablement plus facile de faire la détection d'anomalies. En regardant de plus près les résultats des 4 approches, on remarque que notre approche DA-VAE est rarement meilleure que les méthodes AE ou ACP en aire sous la courbe ROC moyenne. La variabilité autour de la métrique est aussi légèrement plus élevée pour l'approche DA-VAE. Par contre, les résultats sont généralement près des meilleures approches. De plus, l'intervalle avec 2 écart-types autour de la moyenne nous permet de conclure que les autres méthodes ne sont pas significativement meilleurs que DA-VAE. Notre approche montre également les meilleures résultats en aire sous la courbe ROC pour le scénario de test 6, et ce, pour les 3 scénarios de contamination différents. Le scénario de test 6 est celui qui considère le chiffre "6" comme "normal" et tous les autres comme la classe "anormale". Étant donné la nature plus simple des images, les méthodes basées sur la reconstruction (ACP et AE) semblent être celles qui fonctionnent généralement le mieux. On pourrait donc croire que l'utilisation de la représentation latente dans les méthodes ISOF-VAE et DA-VAE n'amènent pas autant de valeur ajoutée que dans le cas d'images réelles comme *ImageNet*. En faisant l'analyse des métriques de précision et de rappel (tableaux 3.7 et 3.8), on peut tirer essentiellement les mêmes conclusions quant aux performances relatives des différentes approches. Encore une fois, les méthodes basées sur la reconstruction semblent légèrement supérieures. Toutefois, l'approche DA-VAE n'est pas significativement inférieure aux autres même en précision et rappel.

Pour illustrer les résultats du modèle selon différents scénarios de contamination, il est possible de réutiliser les mêmes figures que celles décrites à la figure 3.9. Dans la figure 3.16, on peut y voir ces figures générées pour le scénario de test 3 (voir tableau 3.2 pour un rappel des scénarios de test).

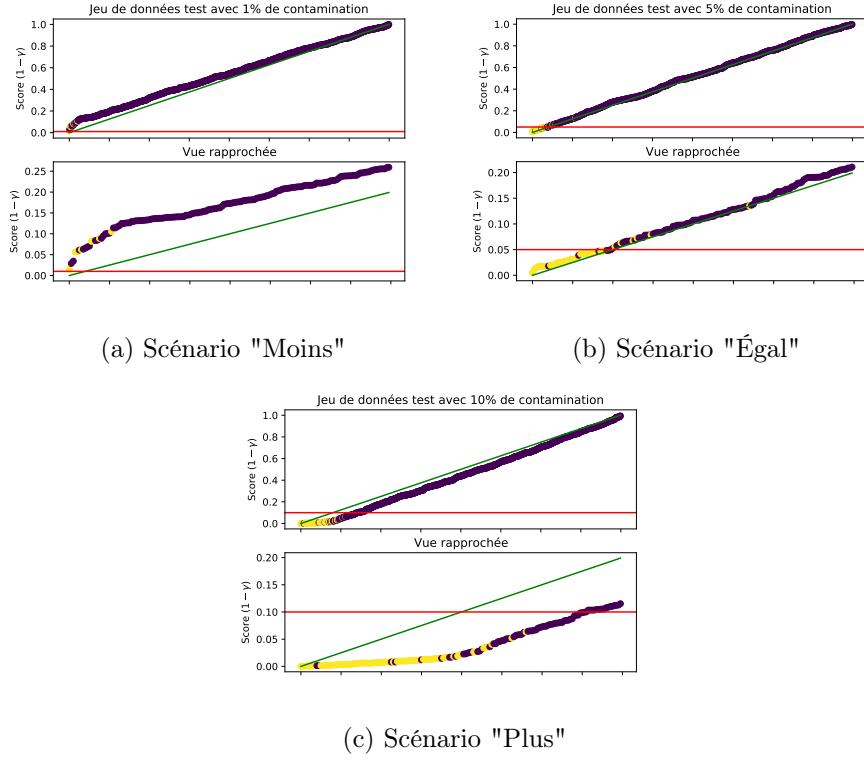


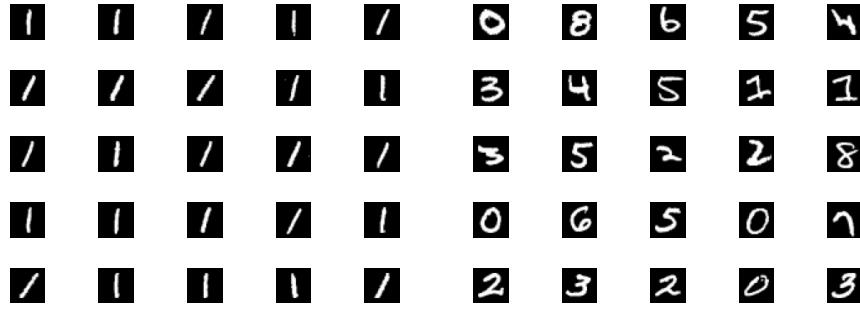
FIGURE 3.16 – Graphiques illustrant les scores d'anomalies selon le scénario de contamination pour le modèle DA-VAE appliqué sur le scénario de test 1 de MNIST. Les points violettes sont des observations que nous connaissons comme "normales" alors que les points jaunes sont des observations que nous connaissons comme "anormales". Dans tous les cas, le niveau de filtration α est défini comme le niveau de contamination dans le jeu de données de test.

Dans les 3 sous-figures à la figure 3.16, on peut voir que les observations "anormales", soit les points jaunes, sont principalement regroupées à gauche de l'axe des x , ce qui est le comportement souhaité. On remarque aussi que la segmentation est plus évidente que dans le cas de la figure 3.9, ce qui nous permet de confirmer que la détection d'anomalies est plus performante sur ce scénario de test du jeu de données MNIST que sur le jeu de données *ImageNet*. Finalement, on peut également remarquer la disposition des observations par rapport à la courbe verte. On peut tirer les mêmes constats que dans la section 3.4.1, mais en ajoutant le fait que les positions des premières observations par rapport à la courbe verte dispositions sont encore plus prononcées dans le cas du jeu de données MNIST.

Analyse des représentations latentes

Dans la section 3.4.1, nous avons fais l'analyse des représentations latentes par l'entremise des statistiques de distance et d'échantillons d'images. Cela nous a permit de conclure que les images "normales" avaient des statistiques de distance élevées alors que les anomalies avaient des statistiques de distance faibles. Il faut donc refaire cette analyse, mais dans le cas de

MNIST. Dans la figure 3.17, on peut voir quelques uns des échantillons d'images du scénario de test 3 nous aidant à tirer notre conclusion. Cette fois-ci, on remarque que les observations ayant des statistiques de distance faibles proviennent de notre classe "normale", soit le chiffre "1". À l'inverse, les observations ayant des statiques de distance élevée proviennent de notre classe "anormale", soit tous les autres chiffres. C'est d'ailleurs le cas pour les 6 différents scénarios de test.

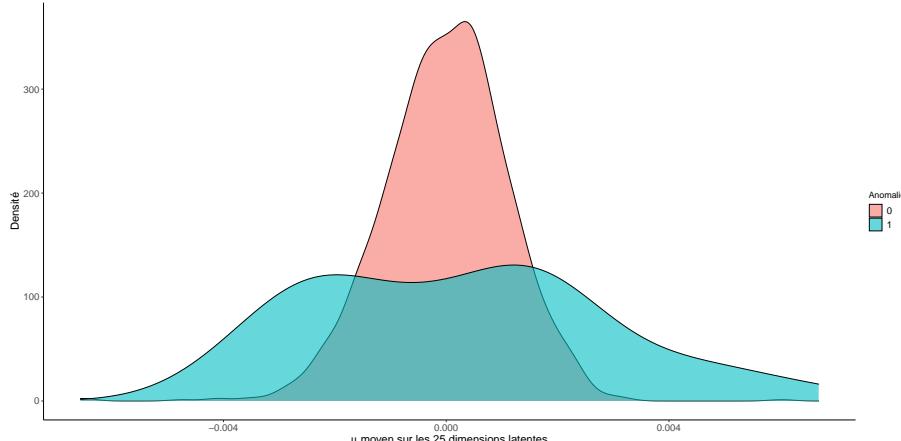


(a) Statistiques de distance faible

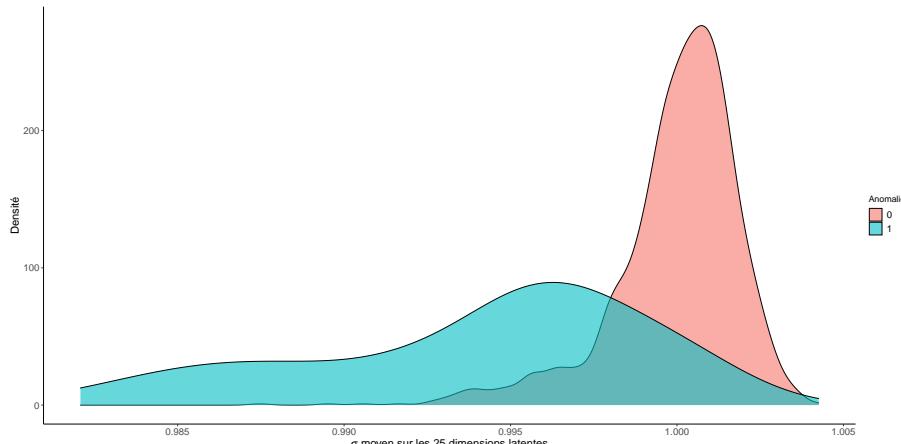
(b) Statistiques de distance élevées

FIGURE 3.17 – Échantillons d'images provenant du jeu de données d'entraînement ayant des statistiques de distance faibles (a) et élevées (b) pour le scénario de test 3 ("Plus") du jeu de données *MNIST*.

De cette manière, il est possible de conclure que nous nous retrouvons plutôt dans le premier scénario décrit (voir les deux scénarios décrits à 2.3.2), soit celui où les représentations latentes des observations "normales" sont plus près de la $N(0, I)$. C'est donc le scénario inverse par rapport aux expérimentations faites sur *ImageNet*. On peut d'ailleurs confirmer cette conclusion dans la figure 3.18, où l'on peut voir que les vecteurs μ et σ moyens des observations "normales" sont plus près des paramètres de moyenne et d'écart-type de la $N(0, I)$.



(a) Distribution des μ moyens



(b) Distribution des σ moyens

FIGURE 3.18 – Moyenne des μ et σ des représentations latentes du jeu de données d’entraînement sur le jeu de données *MNIST*.

Dans la figure 3.19, on peut voir le résultat d’une analyse en composantes principales appliquée sur les vecteurs μ et σ . Cette figure nous permet de visualiser en 2 dimensions, les représentations latentes apprises sur le jeu de données d’entraînement du scénario de test 3 et du scénario de contamination "Plus". Si on compare avec la figure 3.13 qui fait référence au jeu de données *ImageNet*, la séparation des anomalies et des observations "normales" est beaucoup moins évidente. Cependant, les 2 premières composantes principales n’expliquent même pas 25% de la variabilité totale de la représentation latente. En se basant sur cette projection, les "anomalies" semblent être plus éloignées du carré noir, qui représente la projection dans cet espace du point $(\mathbf{0}_m, \mathbf{1}_m)$. Cette conclusion concorde avec l’analyse de la figure 3.18.

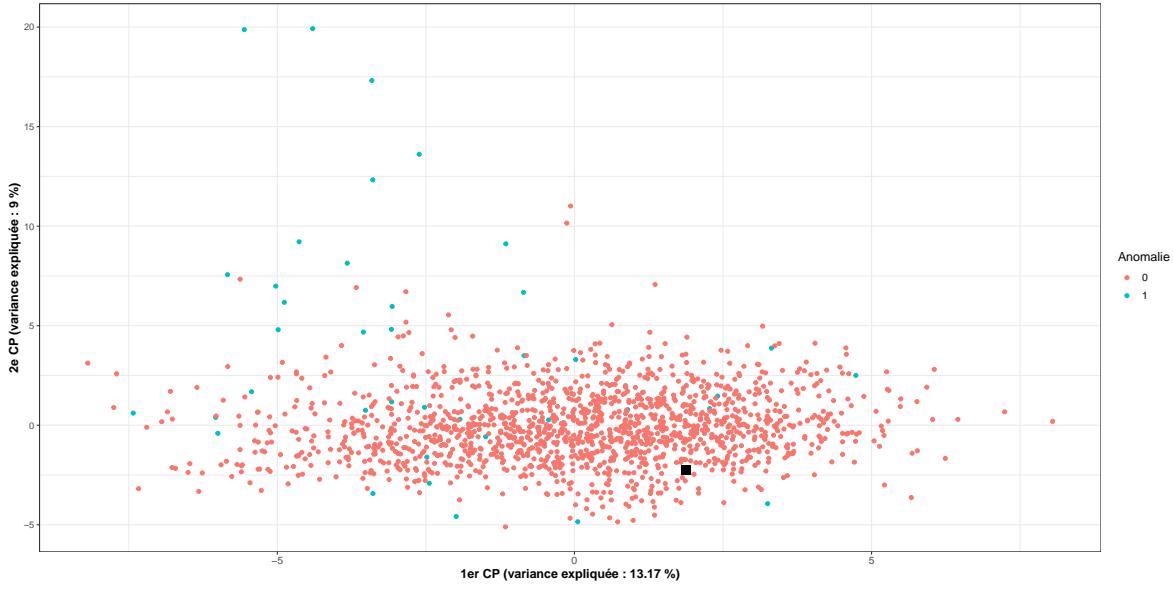


FIGURE 3.19 – Graphique des 2 premières composantes principales réalisé sur les vecteurs μ et σ du jeu de données d’entraînement *MNIST* appliqué sur le scénario de test 3 et le scénario de contamination "Plus".

Analyse de la perte en entraînement

Dans la figure 3.20, on peut voir la composition de la perte à chacune des itérations de l’entraînement. On peut voir que la proportion de la composante de perte de Kullback-Leibler suit le même patron qu’à la figure 3.14. Cependant, la proportion atteint un maximum beaucoup plus élevé que dans le cas du jeu de données provenant de *ImageNet*.

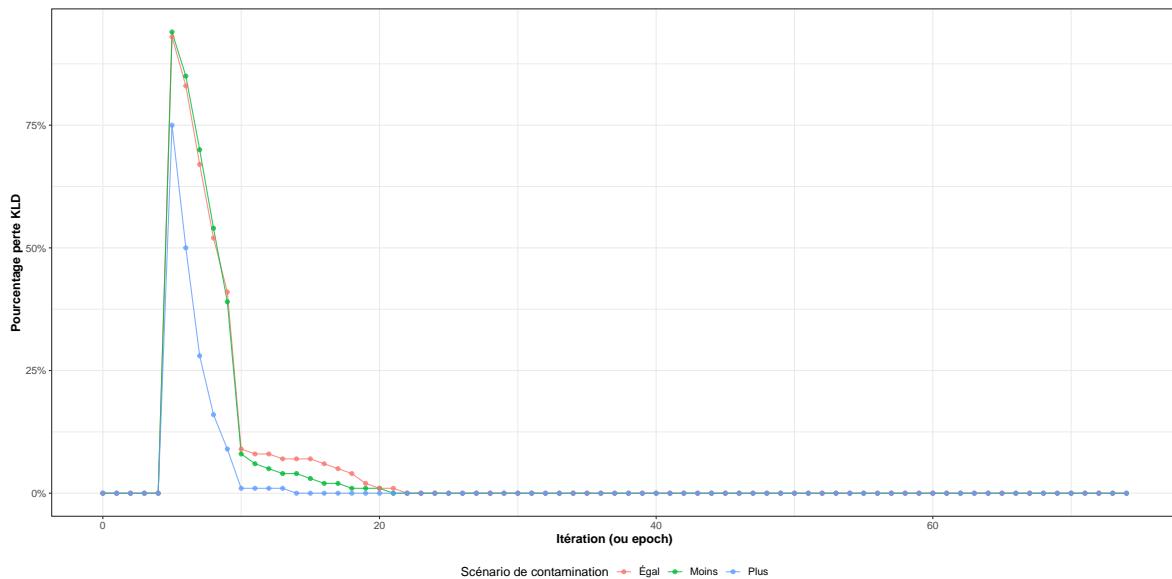


FIGURE 3.20 – Pourcentage du critère de Kullback-Leibler dans la perte totale selon l’itération d’entraînement et le scénario de contamination. Ces résultats sont tirés du scénario de test 3, où le chiffre "1" est considéré comme la classe "normale" et tous les autres sont considérés dans la classe "anormale".

Conclusion

La détection d'anomalies est une tâche qui comporte généralement plusieurs défis en modélisation et en analyse de données. Parmi ces défis, on peut mentionner le débordement entre la classe "normale" et "anormale". Ce débordement peut s'expliquer par le fait qu'une anomalie est par définition un événement qui diffère significativement de la majorité des données (Zimek and Schubert, 2017) et ainsi peut probable. Ensuite, la détection d'anomalies nous amène généralement dans un contexte d'apprentissage non-supervisé. En effet, il est rare qu'on possède des étiquettes nous indiquant si une observation est "anormale" ou pas. Un défi important relié à ce contexte d'apprentissage non-supervisé est que l'analyste ou le chercheur se retrouve souvent dans une situation où il doit déterminer un seuil quelconque afin de pouvoir être en mesure de discriminer les observations "normales" des observations "anormales". L'établissement de ce seuil est généralement basé sur une mesure de distance qui peut être difficile à établir dans certains cas et peut également différer d'une expérience à une autre. Ces défis reliés à la détection d'anomalies deviennent encore plus importants lorsqu'on doit traiter des données complexes, comme par exemple des images. Cette complexité supplémentaire peut se traduire par une forte dimensionnalité des données d'entrées et aussi par une notion de dépendance et de localité entre les pixels voisins d'une image. Dans ce mémoire, nous démontrons la pertinence d'utiliser des autoencodeurs variationnels pour adresser les défis traditionnelles reliés à la détection d'anomalies tout en restant efficace dans une contexte de données complexes comme des images. En bout de ligne, notre approche permet d'identifier les images d'un jeu de données qui dévient de la normalité par rapport à un niveau de filtration, ou d'acceptabilité, intuitif et simple à établir.

Dans les expérimentations réalisées et présentées dans le chapitre 3, nous sommes en mesure de conclure que notre approche produit des résultats performants autant sur des jeux de données d'images réelles que d'images simples. La valeur ajoutée de notre approche par rapport à d'autres méthodes est toutefois beaucoup plus évidente dans le cas d'images réelles. En effet, les performances obtenues par notre approche sur des images simples, comme *MNIST*, ne sont généralement pas significativement meilleures ou pires que d'autres approches. En revanche, les performances en aire sous la courbe ROC, en précision et en rappel obtenues sur des images réelles plus complexes, comme *ImageNet*, sont supérieures aux autres approches comparatives. Nous avons remarqué que l'autoencodeur variationnel permet de représenter les

données complexes dans une forme latente beaucoup plus simple et qui permet aussi de bien discriminer les anomalies des observations "normales".

Notre approche pourrait être utilisée pour plusieurs applications ou situations réelles où l'on cherche à identifier des images "anormales". Prenons l'exemple où l'on bénéficie d'un ensemble d'images propres pour lequel on sait qu'il n'y a pas d'anomalies ou de défectuosités. Supposons que dans le futur, on prévoit recevoir de nouvelles images qui sont supposées représenter la même chose que notre ensemble d'images propres, mais dont certaines de ces nouvelles images sont défectueuses où représentent quelque chose de complètement différent. Nous ne connaissons pas d'avance à quoi ces images défectueuses peuvent ressembler et n'avons aucun exemple d'image de la sorte sur lesquelles on pourrait faire un apprentissage. Notre approche permettrait donc de déterminer lesquelles de ces nouvelles images sont potentiellement défectueuses ou ne semblent pas représenter la même chose que notre ensemble d'images a priori. On pourrait également penser à des exemples d'applications où l'on cherche à nettoyer un jeu de données d'images. En effet, on pourrait valider et confirmer un sous-ensemble d'images propres qui représentent le jeu de données que l'on souhaite avoir. Ensuite, on pourrait utiliser ce sous-ensemble comme jeu de données d'entraînement pour notre approche. Finalement, on pourrait utiliser le reste du jeu de données initial comme jeu de données de test et ainsi valider si des images "anormales" font parties de notre ensemble global. Cela reviendrait donc utiliser notre approche comme technique de nettoyage de données sans avoir à valider toutes les images manuellement.

Dans le futur, il serait intéressant de voir si des variations de l'autoencodeur variationnel pourraient donner des résultats encore plus performants en détection d'anomalies. Parmi les possibilités de variations, il serait intéressant de voir si une autre loi que la $N(0, I)$ pourrait être utilisée comme loi a priori de la représentation latente et voir si cela permettrait d'obtenir une représentation latente qui discrimine encore mieux les anomalies des observations "normales". Ensuite, il serait également intéressant de voir s'il est possible de faire en sorte que les représentations latentes des anomalies se comportent toujours de la même manière par rapport à un point de référence, soit la distribution $N(0, I)$ dans notre cas. En effet, nous avons remarqué que selon la nature des données, les représentations latentes des anomalies se retrouvaient dans certains cas plus près de la $N(0, I)$ et dans certains cas plus éloignées de la $N(0, I)$. Ce constat ajoute une étape manuelle pour mettre en place notre méthodologie, ce qui rend l'approche moins autonome et peut-être plus difficile à appliquer dans la pratique. Au final, nous pouvons conclure que l'autoencodeur variationnel possède des propriétés intéressantes, dont l'apprentissage d'une représentation latente utile, pour faire de la détection d'anomalies sur des données complexes comme des images.

Bibliographie

- Aggarwal, C. C. (2016). *Outlier Analysis*. Springer Publishing Company, Incorporated, 2nd edition.
- An, J. and Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability.
- Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.
- Chandola, V., Banerjee, A., and Kumar, V. (2007). Anomaly detection : A survey.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet : A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.
- Feng, X., Zhang, Y., and Glass, J. (2014). Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition. pages 1759–1763.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.
- Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. (2017). beta-vae : Learning basic visual concepts with a constrained variational framework. In *ICLR*.
- Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786) :504–507.
- Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*.

- Khosla, A., Jayadevaprakash, N., Yao, B., and Fei-Fei, L. (2011). Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO.
- Kingma, D. and Ba, J. (2014). Adam : A method for stochastic optimization. cite arxiv :1412.6980Comment : Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. cite arxiv :1312.6114.
- Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia.
- Krizhevsky, A. and Hinton, G. E. (2011). Using very deep autoencoders for content-based image retrieval. In *ESANN*.
- LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available : http://yann.lecun.com/exdb/mnist*, 2.
- Liu, F. T., Ting, K. M., and Zhou, Z. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422.
- Quattoni, A. and Torralba, A. (2009). Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Parallel distributed processing : Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA.
- Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.
- Statistics, L. B. and Breiman, L. (2001). Random forests. In *Machine Learning*, pages 5–32.
- Zimek, A. and Schubert, E. (2017). *Outlier Detection*, pages 1–5. Springer New York, New York, NY.