Abstract

In this paper, we propose a methodology that aims to detect anomalies among complex data, such as images, and that is based on a confidence level rather than on a problem-specific threshold or value. In order to do that, we demonstrate the usefulness of using variational autoencoders (VAE) to deal with complex data and also to have some kind of representation from which we can apply hypothesis testings. From our experiments, we can show that our approach is able to detect images that are outliers in a given dataset by only specifying a certain confidence level. By using this approach, the anomaly detection becomes feasible for real-world complex images and is also easier to maintain and interpret considering the detection is made on a confidence level rather than on a vague and possibly changing metric.

Table des matières

Liste des tableaux

Liste des figures

<Dédicace si désiré>

Remerciements

<Texte des remerciements en prose.>

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.

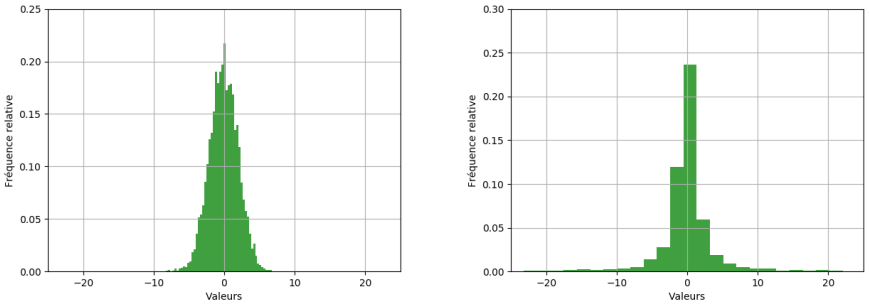
1.1 Les méthodes de détection d'anomalies

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 $X_1, ..., X_N$ avec une moyenne μ et un écart-type σ. La valeur Z pour une observation X_i est donnée par :

$$Z_i = \frac{|X_i - \mu|}{\sigma}. \tag{1.1}$$

Cette valeur Z nous donne en fait le nombre d'écarts-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. Dans la figure 1.1 (image de gauche), on peut voir que cela est plutôt vrai sachant que les données ont été simulées à partir d'une loi $N(0, 2)$. Dans ce cas précis, l'intervalle correspondant à notre règle du pouce est l'intervalle $[-6, 6]$. Cela fonctionne bien puisque notre choix de modèle représente bien les données. Comme contre exemple, 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 (figure 1.1, image de droite), que plusieurs observations ne sont pas considérées comme des anomalies, alors que leur fréquence relative est pourtant très faible.



(a) Distribution normale (b) Distribution standard Cauchy

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. 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.

$$y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \epsilon_i \qquad \forall i \in 1, ..., N \tag{1.2}$$

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

$$\hat{y_i} = \hat{\beta}_0 + \hat{\beta}_1 x_{1,i} + \hat{\beta}_2 x_{2,i} \qquad \forall i \in 1, ..., N \tag{1.3}$$

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 prémisse 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). En effet, 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.

$$Z = XV \tag{1.4}$$

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.

$$\hat{X} = ZV^\top + \boldsymbol{\mu} \tag{1.5}$$

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'interpretabilité. 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 voisin.

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 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 personne 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.

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 décode 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'il peuvent apprendre des variables latentes riches en informations.

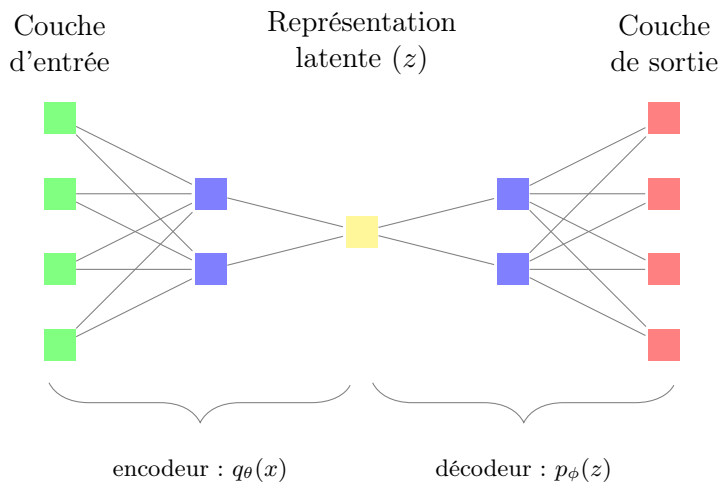


FIGURE 1.2 – Exemple illustrant la structure de base d'un autoencodeur. Dans le cas ci-dessus, on pourrait interpréter le schéma comme un réseau pleinement connecté où les blocs pourraient représentés les neurones et les liens seraient les poids, ou les paramètres du réseau. Le concept s'applique également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

L'intuition derrière les autoencodeurs est essentiellement de reconstruire une entrée x en pas-

sant par 2 fonctions (l'encodeur et le décodeur) apprises par le modèle. 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 :

$$L(x, p_\phi\{q_\theta(x)\}) \tag{1.6}$$

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

$$\Theta \leftarrow \Theta - \epsilon * \frac{\partial L}{\partial \Theta} \tag{1.7}$$

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.

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 entres 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.

La fonction correspondant à l'encodeur, qui transforme l'entrée x vers la représentation latente z, est définie par l'équation 1.8. Dans cette équation, $h(x)$ 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.

$$q_\theta(x) = h\big(x_1 w_{1,1} + x_2 w_{1,2} + b_1\big) \tag{1.8}$$

Pour définir l'encodeur, il est également possible d'utiliser la notation matricielle définie par l'équation 1.9 où $X = [\boldsymbol{x}_1, \boldsymbol{x}_2]$. Pour une seule observation, on peut définir $X^{(i)} = [x_1^{(i)}, x_2^{(i)}]$. Le vecteur de poids $\boldsymbol{w}_1$ est définie comme $\boldsymbol{w}_1 = [w_{1,1}, w_{1,2}]$.

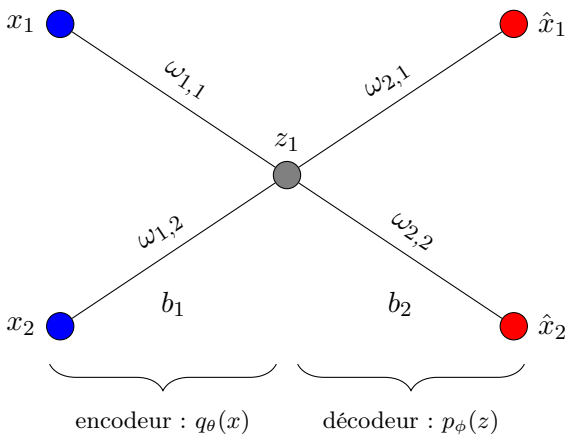


FIGURE 1.3 – Exemple simpliste d'une architecture d'autoencodeur

$$q_\theta(X) = h\big(X * \boldsymbol{w}_1 + b_1\big) \tag{1.9}$$

De la même manière que pour l'encodeur, il est possible de définir de manière plus détaillée la fonction associée au décodeur (équation 1.10).

$$p_\phi(Z) = \begin{cases} p^1_\phi(z) = z * w_{2,1} + b_2 \\ p^2_\phi(z) = z * w_{2,2} + b_2 \end{cases} \tag{1.10}$$

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 1.11 où $\boldsymbol{w}_2 = [w_{2,1}, w_{2,2}]$:

$$p_\phi(Z) = h\big(Z * \boldsymbol{w}_2 + b_2\big) = \hat{X}, \qquad p_\phi : I\!R \to I\!R^2. \tag{1.11}$$

Pour trouver les valeurs optimales des paramètres θ et ϕ du modèle, nous devons définir une fonction de perte (équation 1.6). Supposons que nous voulons minimiser l'erreur de reconstruction seulement. Nous pouvons alors définir notre fonction de perte de la manière suivante :

$$\begin{aligned} L(X, \hat{X}) &= L\big(X, p_\phi\{q_\theta(X)\}\big) \\ &= \frac{1}{2}\sum_{i=1}^{n}[x^{(i)} - p_\phi\{q_\theta(x^{(i)}\}]^{\mathrm{T}}[x^{(i)} - p_\phi\{q_\theta(x^{(i)}\}] \\ &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}\big(x_j^{(i)} - p_\phi^j\{q_\theta(x_j^{(i)})\}\big)^2 \end{aligned} \tag{1.12}$$

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 1.7).

Une fois que l'autoencodeur est entraîné, il est possible d'utiliser l'erreur de reconstruction comme score d'anomalie. En effet, une donné 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 $\boldsymbol{\mu}$ et un vecteur d'écarts-types $\boldsymbol{\sigma}$. Ces deux vecteurs sont ensuite utilisés pour générer la représentation latente à p dimensions, qui est en fait une simulation d'une loi normale multivariée avec les paramètres $\boldsymbol{\mu}$ et $\boldsymbol{\sigma}$. 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 $\boldsymbol{\mu}$ et $\boldsymbol{\sigma}$ 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 $\boldsymbol{\mu}$ et $\boldsymbol{\sigma}$, 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 ($\boldsymbol{\mu}$ et $\boldsymbol{\sigma}$). La représentation latente z est ensuite générée en simulant d'une loi normale multivariée avec les paramètres $\boldsymbol{\mu}$ et $\boldsymbol{\sigma}$. 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 ($p_\phi(z)$).

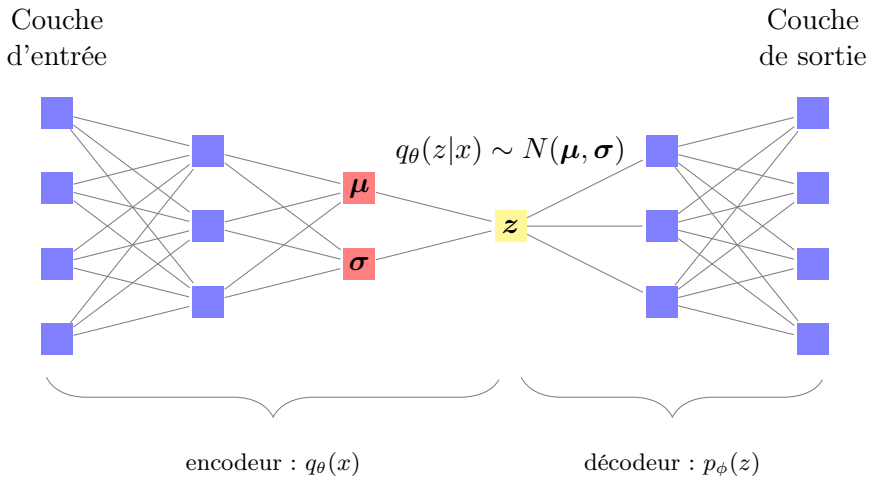


FIGURE 1.4 – Structure de base d'un autoencodeur variationnel. La représentation latente z est générée en simulant de la loi $q_\theta(z|x)$, qui suit une loi normale multivariée avec les paramètres μ et σ calculés par le réseau. C'est de là que vient la composante stochastique du modèle.

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 :

$$L(x, p_\phi\{q_\theta(x)\}) + D_{KL}\big[q_\theta(z|x)||p(z)\big], \tag{1.13}$$

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

$$
\begin{aligned}
D_{\mathrm{KL}}(P||Q) &= \int_{-\infty}^{\infty} p(x)\log\Big(\frac{p(x)}{q(x)}\Big) \\
&= \int_{-\infty}^{\infty} p(x)(\log(p(z)) - \log(q(z))).
\end{aligned}
\tag{1.14}
$$

Dans le cas précis de l'autoencodeur variationnel, la distribution $q_\theta(z|x)$ est une loi normale multivariée de paramètres $\boldsymbol{\mu}$ et $\boldsymbol{\sigma}$. La loi $p(z)$, ou la loi à priori, est une loi normale multivariée standard ou $N(0, I)$. 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 $N(\boldsymbol{\mu}, \boldsymbol{\sigma})$ et une loi $N(0, I)$ à k dimensions, l'équation 1.14 se développe comme suit :

$$
\begin{aligned}
D_{\mathrm{KL}}(N_{\boldsymbol{\mu},\boldsymbol{\sigma}}||N_{0,I}) &= \int_{-\infty}^{\infty} N_{\boldsymbol{\mu},\boldsymbol{\sigma}}(x)\big\{\log(N_{\boldsymbol{\mu},\boldsymbol{\sigma}}(x)) - \log(N_{0,I}(x))\big\} \\
&= \frac{1}{2}\big\{\mathrm{tr}(\boldsymbol{\sigma^2}) + \boldsymbol{\mu^T\mu} - k - \log\,\det(\boldsymbol{\sigma^2})\big\}.
\end{aligned}
\tag{1.15}
$$

.

La trace calculée sur $\boldsymbol{\sigma^2}$ 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 1.15 comme :

$$
\begin{aligned}
D_{\mathrm{KL}}(N_{\boldsymbol{\mu},\boldsymbol{\sigma}}||N_{0,I}) &= \frac{1}{2}\Big\{\sum_k \sigma_k^2 + \sum_k \mu_k^2 - \sum_k 1 - \log\prod_k \sigma_k^2\Big\} \\
&= \frac{1}{2}\Big\{\sum_k \sigma_k^2 + \sum_k \mu_k^2 - \sum_k 1 - \sum_k \log(\sigma_k^2)\Big\} \\
&= \frac{1}{2}\sum_k \Big\{\sigma_k^2 + \mu_k^2 - 1 - \log(\sigma^2)\Big\}.
\end{aligned}
\tag{1.16}
$$

Pour des raisons de stabilité numérique, on utilise souvent la version logarithme de la matrice $\boldsymbol{\sigma^2}$, ce qui nous permet de réécrire l'équation 1.16 comme suit :

$$
D_{\mathrm{KL}}(N_{\boldsymbol{\mu},\boldsymbol{\sigma}}||N_{0,I}) = \frac{1}{2}\sum_k \Big\{\exp(\sigma_k^2) + \mu_k^2 - 1 - \sigma^2\Big\}. \tag{1.17}
$$

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 un vecteur de 1. 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 :

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} * \boldsymbol{\epsilon}$$

où $\boldsymbol{\epsilon} \sim N(0, I)$. En bref, cela signifie que la couche z, illustrée dans la figure 1.4, est générée à partir de deux couches de paramètres $\boldsymbol{\mu}$ et $\boldsymbol{\sigma}$ 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 $\boldsymbol{\mu}$ et $\boldsymbol{\sigma}$ seulement et ignorer $\boldsymbol{\epsilon}$.

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 ou 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, définit 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 :

$$L(x, p_\phi\{q_\theta(x)\}) + \beta \times D_{KL}\big[q_\theta(z|x)||p(z)\big]. \tag{1.18}$$

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 la

représentation latente est *disentangled*. 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, l'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.

1.4 Les tests d'hypothèses

Les tests d'hypothèse sont une méthode d'inférence statistique. L'objectif est de tester si les données contiennent assez d'évidence pour rejeter une hypothèse nulle (H_0) en faveur d'une hypothèse alternative (H_1). Dans le contexte de détection d'anomalie, nous pourrions utiliser ce cadre d'inférence pour tester si une certaine observation provient d'une population autre que celle attendue :

$\boldsymbol{H_0}$: x_i provient de la population P
$\boldsymbol{H_1}$: x_i ne provient pas de la population P

Une fois que nous avons défini une structure de test, il faut se définir un cadre précis nous

permettant de mettre en application ce test. Cela passe généralement par une hypothèse à priori se décomposant en deux parties : une distribution attendue et une statistique de test découlant de cette distribution. Avec ces deux composantes, on peut calculer une valeur-p pour le test en question. La valeur-p est définie comme la probabilité, sous l'hypothèse nulle, d'observer la statistique de test de la distribution attendue. Quand la valeur-p est petite, cela signifie qu'il est peu probable d'observer l'observation sous l'hypothèse nulle. L'hypothèse nulle est généralement rejetée lorsque la valeur-p est plus petite qu'un certain seuil α. Ce seuil est également appelé le niveau de confiance. Lorsque l'hypothèse nulle est vraie et que la distribution testée est continue, alors la distribution de cette valeur-p lors de tests répétés sur des échantillons indépendants est distribuée de manière uniforme sur l'intervalle [0, 1].

Dans ce projet, un de nos objectif est d'exploiter ce cadre d'inférence statistique en utilisant un niveau de confiance, plutôt qu'une métrique quelconque pour détecter les anomalies. Ainsi, nous pouvons avoir un niveau de certitude quant à la décision de catégoriser une observation comme une anomalie. De plus, le niveau de confiance α, qui nous permet de prendre une décision, est une métrique beaucoup plus intuitive et facile à déterminer comparativement à une mesure de distance quelconque.

Avec ce survol des méthodes de détection d'anomalies, des autoencodeurs et des tests d'hypothèses, 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 sera transformée vers une métrique que nous pourrons utiliser dans un le cadre d'un test d'hypothèse. Ce test d'hypothèse nous permet donc de détecter des anomalies avec un niveau de confiance.

2.1 Les hypothèses de l'approche

Dans notre approche, nous supposons que nous avons accès à un jeu de données qui contient presque qu'entièrement des observations normales. En d'autres mots, il n'y pratiquement pas d'anomalies. Par contre, nous anticipons des entrées anormales dans le futur et nous voulons les détecter. Voici 2 exemples de situation où ce genre de contexte pourrait être plausible :

1. Une compagnie d'assurance propose maintenant à leurs assurés de prendre eux-mêmes des photos de leur véhicule accidenté et d'envoyer celles-ci à l'assureur. La compagnie s'attend à ce que les assurés puissent commettre des erreurs de manipulation en prenant les photos ou bien puissent envoyer les mauvaises photos. De son côté, l'assureur possède des photos de véhicules accidentés, prises par son personnel d'estimateurs, mais aucune photo erronée. Le jeu de données d'entraînement ne possède théoriquement aucune anomalie. Par contre, des anomalies sont à prévoir dans le futur et on veut les identifier.
2. Une entreprise manufacturière qui produit des pièces de voiture vient de se procurer un nouveau système qui automatise davantage la production. L'entreprise veut faire un suivi des pièces produites par ce nouveau système via une caméra qui inspecte les produits finaux. L'entreprise possède déjà plusieurs images de pièces qui étaient produites avant l'acquisition du nouveau système. Le résultat final produit par ce nouveau système doit être identique à l'ancien. Par contre, il est difficile de prévoir de quoi auront l'air des

défectuosités produites par ce nouveau système. Le jeu de données d'entraînement est donc constitué de plusieurs exemples normales, mais aucune anomalie.

2.2 Description de l'approche

Considérant que nous avons accès à un jeu de données ayant les caractéristiques décrites à la section précédente, nous proposons d'entraîner un autoencoder variationnel pour apprendre les caractéristiques, ou la distribution, de cette population "normale". Cette distribution apprise sera essentiellement contenu dans la représentation latente du VAE entraîné, ou plus spécifiquement dans les couches μ et σ du réseau. Comme nous l'avons vu dans la section 1.3, les VAE ont la particularité d'avoir une composante de perte associée à cette représentation latente, s'assurant ainsi que celle-ci s'approche d'une distribution à priori, soit une $N(0, I)$ dans notre cas. Une fois que l'autoencodeur est entraîné, nous pouvons utiliser la partie encodeur du réseau pour transformer chacune des instances de notre jeu de données d'entraînement vers des vecteurs μ et σ. Ces représentations latentes, encodées sous forme de vecteurs, contiennent l'information qui devrait permettre de savoir si une nouvelle instance partage ou non cette même information. Pour parvenir à faire cette conclusion sur une nouvelle instance, nous devons comparer l'information contenue de la représentation latente de cette instances avec toutes les représentations latentes de l'ensemble d'entraînement. Il est plus pratique d'agréger les représentations notre population d'entraînement en une seule représentation globale. Une manière d'y arriver est de calculer un vecteur $\bar{\mu}$ et un vecteur $\bar{\sigma}$, qui sont simplement une moyenne sur la population d'entraînement. Supposons que notre jeu de données d'entraînement contient n observations et que nous encodons celles-ci vers des représentations latentes à k dimensions, l'élément i des vecteurs $\bar{\mu}$ et $\bar{\sigma}$ sera donné par :

$$\bar{\mu}_i = \frac{1}{n} \sum_{l=1}^{n} \mu_i^{(l)}$$
$$\bar{\sigma}_i = \frac{1}{n} \sum_{l=1}^{n} \sigma_i^{(l)}$$

Étant donné que les vecteurs $\bar{\mu}$ et $\bar{\sigma}$ sont essentiellement composé d'instances normales, on peut s'attendre que les vecteurs μ et σ de nouvelles instances normales soient relativement près de ces vecteurs moyen. À l'inverse, on peut s'attendre que ces 2 mêmes vecteurs, pour des instances anormales, ou des anomalies, soient plutôt loin de ces vecteurs moyens. Maintenant, comment savoir si la représentation latente d'une nouvelle instance est proche ou éloignée de cette représentation agrégée ? Étant donné que les vecteurs μ et σ représentent les paramètres d'une loi normale multivariée, nous pouvons utiliser la distance de Kullbach-Leibler pour quantifier le degré de similitude entre la distribution moyenne et la distribution μ et σ de la nouvelle instance. Cette même distance est d'ailleurs utilisée lors de l'optimisation

pour calculer la composante de perte associée à la représentation latente. Finalement, il est possible d'utiliser les distances de KL calculées sur l'ensemble d'entraînement pour savoir si une nouvelle distance est élevée ou non. Cela devient possible étant donné que notre ensemble d'entraînement contient presque uniquement des observations normales, donc des distances de KL qui devraient être petites. On peut d'ailleurs utiliser ces distances ordonnées pour avoir une mesure de probabilité d'une nouvelle instance. La méthodologie proposée pour calculer cette mesure de probabilité, ou cette valeur-p est résumé dans l'algorithme 1.

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

Input : Ensemble de données d'entraînement sans anomalie $x^{(j)}, j = 1, ..., m$,
Ensemble de données de test avec anomalies $y^{(i)}, i = 1, ..., n$
Output : Valeurs-p pour chaque instance de test $p^{(i)}, i = 1, ..., n$
θ, $\phi \leftarrow$ paramètres de l'encodeur ($q_\theta(x)$) et du décodeur ($p_\phi(z)$) du VAE entraîné;
for *j=1 to m* **do**
    $\mu^{(j)} = p_\theta(x^{(j)})["mu"]$;
    $\sigma^{(j)} = p_\theta(x^{(j)})["sd"]$;
end
$\bar{\mu} = \frac{1}{m}\sum_{k=1}^{m} \mu^{(k)}$;
$\bar{\sigma} = \frac{1}{m}\sum_{k=1}^{m} \sigma^{(k)}$;
for *j=1 to m* **do**
    $kl\_train^{(j)} = kl\_distance(\mu^{(j)}, \sigma^{(j)}, \bar{\mu}, \bar{\sigma})$
end
$kl\_sorted = sort(kl\_train)$;
for *i=1 to n* **do**
    $\mu^{(i)} = p_\theta(y^{(i)})["mu"]$;
    $\sigma^{(i)} = p_\theta(y^{(i)})["sd"]$;
    $kl\_test^{(j)} = kl\_distance(\mu^{(i)}, \sigma^{(i)}, \bar{\mu}, \bar{\sigma})$;
    $p^{(i)} = rank_{kl\_sorted}(kl\_test)$ / m
end
return $\boldsymbol{p}$

Dans l'approche que nous proposons, la valeur-p est calculée à partir d'une distribution empirique, soit les distances de Kullbach-Leibler de toutes les instances du jeu de données d'entraînement. Au final, nous testons empiriquement si une nouvelle observation semble provenir de la population d'entraînement :

$$\boldsymbol{H_0} : y^{(i)} \text{ provient de la population } X$$
$$\boldsymbol{H_1} : y^{(i)} \text{ ne provient pas de la population } X$$

Puisque nous supposons que X, soit l'ensemble de données d'entraînement, est composé en grande partie d'observations normales, nous pouvons réécrire notre test comme étant :

$$\boldsymbol{H_0} : y^{(i)} \text{ n'est pas une anomalie}$$
$$\boldsymbol{H_1} : y^{(i)} \text{ est une anomalie}$$

Finalement, une fois que nous avons notre test statistique de définit, nous pouvons utiliser les valeurs-p calculées sur l'ensemble de test et conclure avec un niveau de confiance α si une instance est une anomalie (voir algorithme 2).

Algorithme 2 : Algorithme de prise de décision

Input : Valeurs-p pour les instances de test $p^{(i)}, i = 1, ..., n$,
niveau de confiance α
Output : indicateurs d'anomalies $o^{(i)}, i = 1, ..., n$
for *i=1 to n* **do**
  if $p^{(i)} < \alpha$ **then**
    $o^{(i)} = vrai$
  else
    $o^{(i)} = faux$
  end
end
return ***o***

2.3 Gestion des données complexes

Un des objectifs de notre approche est de faire la détection d'anomalies parmi des données qui sont complexes, comme par exemple des images réelles. Sachant ce niveau de complexité, il faut que la méthodologie soit adaptée à ce contexte. Tout d'abord, il est pertinent de mentionner que le choix d'utiliser des réseaux de neurones, plus particulièrement des autoencodeurs, est étroitement liée à ce concept de complexité. En effet, les réseaux de neurones ont le potentiel de stocker beaucoup d'informations au travers des nombreux paramètres du réseaux. De plus, les réseaux à convolutions sont également bien adaptés aux domaine de l'imagerie en prenant en compte l'information de manière locale dans l'image. Dans la prochaines sous-sections, nous allons présenter d'autres spécifications de l'approche qui permettent de prendre en compte cette complexité.

2.3.1 L'utilisation de représentations latentes

Lorsqu'on parle de données complexes, on parle souvent de réduction de dimensionnalité. En effet, des données à haute dimensionnalité viennent généralement compliqué la tâche d'ap-

prentissage, alors que les nombreuses dimensions augmente la distance entre les observations. Ce concept est d'autant plus important dans un contexte de détection d'anomalie, où on se rapporte souvent à une mesure de distance ou bien à une erreur de reconstruction. Prenons l'exemple où on doit détecter des anomalies parmi des images de dimensions 128 par 128. Supposons que nous utilisons l'approche avec un VAE décrite plus tôt et que nous encodons les données vers une représentation latente à 25 dimensions. En utilisant une distance de Kullbach-Leibler sur la représentation latente, on calcule une distance basée sur 2 vecteurs à 25 dimensions pour obtenir un score d'anomalie. Pour bien visualiser l'effet du fléau de la dimensionnalité, supposons que nous aurions plutôt décider d'utiliser l'erreur de reconstruction, basée sur une erreur quadratique moyenne, comme score d'anomalie. Dans ce cas-ci, nous aurions donc calculé notre erreur quadratique moyenne en fait la différence au carré entre les valeurs de chaque pixel prédite par le modèle et de l'image originale. Ensuite, nous aurions pris la moyenne de ces erreurs. Sachant que nous avons à l'origine des images 128 par 128 pixels, cela veut dire que notre score d'anomalie serait basé sur 16 384 dimensions. On comprend bien que le score basé sur la représentation latente a le potentiel d'être moins affecté par la haute dimensionnalité des données.

2.3.2 Fonction de perte orientée sur le contenu

Dans la section précédente, nous avons mentionné l'impact de la haute dimensionnalité dans le calcul du score d'anomalie. Un autre aspect qui peut être fortement impacté par cette complexité est la fonction de perte. Cette fonction de perte est primordiale lors de l'apprentissage alors que c'est elle qui guidera l'optimisation des paramètres par la descente du gradient. La fonction de perte de l'autoencodeur variationnel est donnée par l'équation 1.13, où une des deux composantes de cette perte est associée à l'erreur de reconstruction. Dans sa forme la plus simple, cette erreur de reconstruction est définit pixel par pixel. Dans le cas de l'erreur quadratique moyenne, cette composante de perte peut être définit comme :

$$L(x, p_\phi\{q_\theta(x)\}) = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - p_\phi\{q_\theta(x^{(i)})\})^2 \tag{2.1}$$

où n représente le nombre de d'éléments de l'entrée x, ou plus spécifiquement le nombre de pixels dans le cas d'une image.

Le genre de fonction de perte définit à l'équation 2.1 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 d'arrière-plan dans le coin de l'image, que les pixels centrales qui pourrait nous donner des informations plus cruciales sur le contenu de l'image. Dans un contexte où on cherche à trouver des anomalies au niveau de l'image entière, et non des anomalies dans une zone de l'image, il devient pertinent de trouver une manière d'accorder plus d'importance au contenu

global de l'image. Pour se faire, on utilise un concept qui s'appelle *perceptual optimization*. Dans ce type d'optimisation, on calcul 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* ou *ResNet*, pré-entraînés sur *ImageNet*. Cette approche est d'ailleurs utilisée pour transformer le style d'une image, par exemple pour donner le style van Gogh à une photo quelconque (Johnson et al. (2016)). La figure 2.1 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.

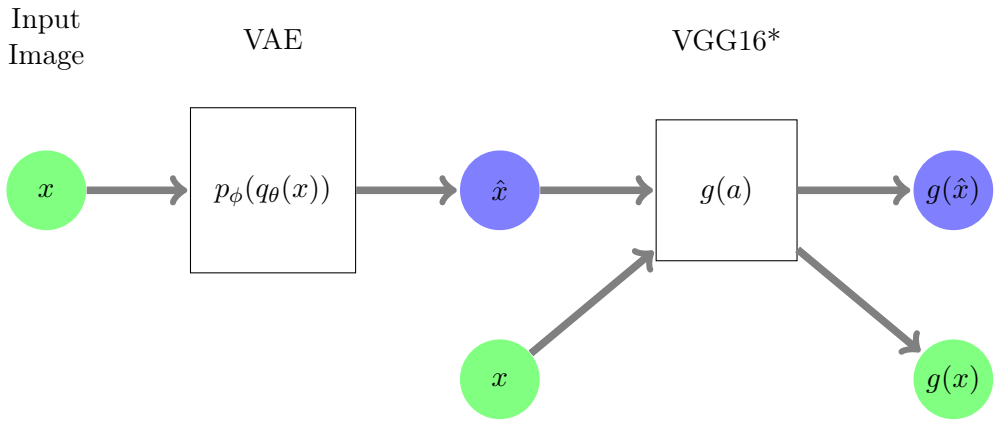


FIGURE 2.1 – Figure montrant le mécanisme derrière le*perceptual optimization*. Au lieu de calculer la perte entre x et $\hat{x}$, la perte est calculée entre $g(x)$ et $g(\hat{x})$. La fonction $g(a)$ permet d'extraire les valeurs d'une couche spécifique du réseau *VGG16* pré-entraîné sur *ImageNet*.

Étant donné que le réseau correspondant à la fonction $g(a)$ dans la figure 2.1 est pré-entraîné sur plusieurs images réelles, cela nous permet d'extraire une couche qui contient de l'information à plus bas niveau comme des lignes, des formes, des couleurs, etc. Plus la couche sélectionné est près de l'entrée du réseau, plus l'information sera de simple, par exemple des lignes dans l'image. 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. L'erreur de reconstruction qui était définit à l'équation 2.1, est maintenant définit à l'équation .

$$L(x, p_\phi\{q_\theta(x)\}) = \frac{1}{n}\sum_{i=1}^{n}(g(x^{(i)}) - g(p_\phi\{q_\theta(x^{(i)})\})^2) \tag{2.2}$$

L'objectif globale de cette approche 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 entres autres d'orienter l'apprentissage vers le contenu à 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. Dans ce contexte, il est d'autant plus pertinent de mettre l'accent sur le contenu.

2.4 Avantages du test d'hypothèse

Parler du fait que c'est simple de definir un level of significance versus trouver une metric quelconque.

Chapitre 3

Expérimentations

3.1 Jeux de données

3.1.1 MNIST

3.1.2 ImageNet

3.2 Modèles testés

3.3 Détails d'entraînement

3.4 Discussion

Conclusion

<Texte de la conclusion. Une thèse ou un mémoire devrait normalement se terminer par une conclusion placée avant les annexes, le cas échéant. La conclusion est traitée comme un chapitre normal, sauf qu'elle n'est pas numérotée.>

Annexe A

<Titre de l'annexe>

<Texte de l'annexe.>

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.

Chandola, V., Banerjee, A., and Kumar, V. (2007). Anomaly detection : A survey.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

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*.

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*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. cite arxiv :1312.6114.

Zimek, A. and Schubert, E. (2017). *Outlier Detection*, pages 1–5. Springer New York, New York, NY.