Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université SAAD Dahleb_BLIDA

Faculté des sciences

Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en informatique

OPTION : Ingénierie des logiciels

Le thème

Proposition d'une approche de génération des formes à partir d'une description textuelle

Présenté par :

- MOHAMMED AZIZI Kamelia.
- SERIR Fella

Membre de jury

Encadreur : **Mr**.A.H.KAMECHE

Président du jury :**Mme**.I.CHERFA

Examinatrice : **Mme**.M.FARAH

Année universitaire 2019 /2020.

Résumé

La conception des formes 3D sur ordinateur se caractérise par une diversité d'attributs, de ce fait, elle était toujours considérée comme une tâche difficile aux concepteurs, même s'il était possible de la faire, elle ne pourra pas être réalisée ni en peu de temps, ni avec des logiciels moins coûteux .C'est la raison pour laquelle nous avons pensé à créer un système intelligent et convivial pour résoudre ces problèmes.

L'objectif clé de ce travail s'articule beaucoup plus sur la génération des formes 3D à partir des descriptions textuelles en utilisant des réseaux de neurones génératifs GAN et leurs variantes. Les résultats obtenus ont montré que le choix des paramètres et le grand nombre d'échantillons de chaque jeu de données influent sur la qualité des résultats rendus et aident le modèle à s'entraîner et d'apprendre plus d'une forme 3D.

Mots clé: Apprentissage automatique, descriptions textuelles, formes 3D,texte-en-forme,GAN

Abstract

The design of 3D shapes on a computer is characterized by a diversity of attributes, therefore it was always considered a difficult task for designers, even if it was possible to do it, it cannot be carried out in a short time, nor with less expensive software. That's why we thought about creating a smart, user-friendly system to solve these problems.

The key objective of this work is much more focused on generating 3D shapes from textual descriptions using GAN generative neural networks and their variants. The results obtained showed that the choice of parameters and the large number of samples from each dataset affect the quality of the results rendered and help the model practice and learn more than a 3D shape.

Keywords: Machine learning, text descriptions, 3D shapes, text-to-shape,GAN.

الملخص

يتميز تصميم الأشكال ثلاثية الأبعاد على الكمبيوتر بتنوع السمات ، لذلك كان يعتبر دائمًا مهمة صعبة للمصممين ، حتى لو كان من الممكن القيام بذلك ، فلا يمكن انجازها ، لا في وقت قصير ، ولا باستخدام برامج أقل تكلفة ، ولهذا السبب فكرنا في إنشاء نظام ذكي وسهل الاستخدام لحل هذه المشكلات.

يدور الهدف الرئيسي لهذا العمل بشكل أكبر على توليد الأشكال ثلاثية الأبعاد من الأوصاف النصية باستخدام الشبكات العصبية التوليدية لـ GAN ومتغيراتها. أظهرت النتائج التي تم الحصول عليها أن اختيار العوامل والعدد الكبير للعينات من كل مجموعة بيانات يؤثر على جودة النتائج المعروضة ويساعد النموذج على تدريب وتعلم أكثر من شكل ثلاثي الأبعاد.

الكلمات الرئيسية: تعلم الآلة ، أوصاف النص ، والأشكال ثلاثية الأبعاد ، وتحويل النص إلى شكل, شبكات التوليد الخصيمة.

Liste des abréviations

Abréviation	Description complète
CNN	Convolutional neural network. Réseau de neurone Convolutionnel .
FNN	Feed-forward neural network. Réseau de neurones à réaction.
RNN	Recurrent neural network. Réseau neuronal récurrent.
GRU	Gated Recurrent Unit. Unité récurrente à portes..
LSTM	Long Short-Term Memory. Longue courte terme mémoire.
GAN	Generative adversarial networks. Réseaux antagonistes génératifs.
CGAN	Conditional generative adversarial network. Réseau Antagoniste Génératif Conditionnel.
WGAN	Wasserstein generative adversarial network. Réseau Antagoniste Génératif Wasserstein .
TALN	Automatic Natural Language Processing. Traitement automatique du langage naturel.
TIC	Information and communication technologies. Technologies de l'information et de la communication.
NLP	Natural Language Processing. Traitement du langage naturel.

REMERCIEMENT

Aujourd'hui, suite à la clôture de notre parcours universitaire, nous tenons à noter que cette année fut la plus marquante de toutes.

Nous remercierons en premier lieu le **Dieu**, le tout puissant de nous avoir donné le courage et la patience pour faire notre travail, en suite nos parents.

Nous adressons aussi notre sincère sentiment de gratitude à notre encadreur **Mr KAMECHE Abdellah Hicham** pour ses conseils et encouragement..

Nous souhaitons exprimer nos remerciements à tous les enseignants du département informatique de Blida qui ont assuré notre formation durant nos cinq années d'études et les membres de spécialité **IL**.

Nous tenons également à remercier toute personne ayant contribué de près ou de loin à la concrétisation de ce mémoire.

Merci

Kamelia et Fella

DEDICACE

J'ai le grand plaisir de dédier ce modeste travail

À ma chère **Maman**, qui me donne toujours l'espoir de vivre et qui n'a jamais cessé de prier pour moi.

À mon cher **Père**, pour ses encouragements, son soutien, surtout

pour son amour et son sacrifice que rien n'entrave le déroulement de mes études .

Je dédie ce travail à mon frère **Yacine** et ma sœur **Dr**. Yasmine pour se conseille durant toute la période de travail.

À mes meilleures amies **Mohammed El Hadj Khaoula** et **Mohammed Safia**

qui m'a toujours encouragé, et je leur souhaite plus de bonheur et de succès dans leur vie.

Un grand remerciement à mon binôme et ma sœur Fella qui a été présent avec moi

durant toute cette période difficile afin de faire du bon travail.

À tous mes camarades de l'option **IL** qui ont été comme ma deuxième famille.

Et enfin à tous les étudiants de département d'informatique.

Kamelia

DEDICACE

Avec tout respect et amour je dédie ce modeste travail

À mes chers parents

Pour leur amour, leur soutien, pour tous les efforts consentis pour m'assurer une bonne éducation et pour leur assistance et leur présence dans ma vie.

A mon binôme Kamelia,

Mon amie et ma main droite ; j'espère à elle un bon avenir et une bonne continuité dans sa vie et une vie pleine de paix et de bonheur.

A mes amies

Chaïma, Nour El Houda, Sara

Pour les encouragements et pour tous les bons moments qu'on a vécus ensemble.

J'espère que notre amitié durera éternellement.

Et plusieurs d'autres dont je reconnais le bien.

Et A

ZOHEIR EL HOUARI

Pour tout son aide.

Et enfin à tous les étudiants de département d'informatique.

Fella

Table des matières

Liste des figures

❖ Chapitre 1

❖ Chapitre 3

❖ Chapitre 4

Liste des Tableaux

❖ Chapitre 1

❖ Chapitre 2

❖ Chapitre 4

Liste des Graphes

Chapitre 4

Introduction générale

Durant les dernières années et avec l'évolution des technologies de l'information et de la communication **(TIC)**, l'intelligence artificielle « **IA** » est devenue un élément très important au niveau de la création des applications de haute qualité avec une nouvelle méthode d'implémentation basée sur le machine Learning. Parmi ces techniques, on trouve des applications diverses et variées permettant, par exemple :la reconnaissance et la génération : (des formes, des images...) et même pour le cas contraire.

À travers les recherches qu'on a faite, on a choisi ce thème car nous voulons rencontrer dans le domaine d'intelligence artificielle avec l'utilisation de langage python qui a ajouté un nouveau souffle à nos connaissances et spécialité.

L'objectif principal de notre travail, permet de créer une application qui fonctionne et interagir comme le cerveau d'un être humain, capable de détecter et comprendre les mots pertinents situés dans les descriptions textuelles et transformés en forme 3D (tables et des chaises) coloré.

Ce dernier est dédié spécialement pour aide les architectes et les designers surtout pour l'aménagement des maisons, en plus il est également utilisé dans les entreprises de fabrication des meubles pour réaliser plus de formes selon les spécifications souhaitées par le client.

L'approche de génération des formes 3D elle n'est pas dédiée uniquement dans les deux cas précédents, mais elle utile aussi pour la création des jeux vidéo. Le cas d'usage de system est varié selon le type des formes utilisé (table, chaise, humain...).

L'approche est utile et intéressante, mais la conception des formes en 3D (3 dimensions) sur ordinateur n'est pas une tâche facile à faire, car cette dernière nécessite des concepteurs qualifiés dans leur domaine et des logiciels de modélisation 3D et CAO (Conception assistée par ordinateur) qui sont très couteaux tels que : (Maya, Blender, 3 ds max …). Parfois les concepteurs trouvent des difficultés à réaliser la forme 3D qui regroupe plusieurs caractéristiques en même temps, comme :

La catégorie (tables, chaise …),la couleur (rouge, bleu …),le matériau (bois, verre, fer …).

Même s'il était possible, ils ne pourraient pas réaliser un grand nombre des formes dans des délais minimisés.

Donc la présente étude vise à répondre à une problématique générale qui est de savoir est-ce qui il est possible d'exploiter ces techniques pour concevoir un système qui remplace les humains pour créer des formes tridimensionnelles colorées avec plusieurs caractéristiques à travers les descriptions textuelles ?

Le système en question doit

Détecter et comprendre les mots pertinents d'une description textuelle donnée en utilisant les outils de NLP et de générer et d'associer des formes 3D à leurs descriptions correspondantes. Et même comprendre et interpréter des formes 3D soumises sous format de voxels.

Si on a réussi à créer une application avec les conditions proposées précédentes :
Comment il le système faire pour détecter et comprendre les mots pertinents d'une description textuelle donnée et transformée en forme 3D colorée ?
Est-ce qu'il existe des méthodes qui comprend le sens de mot d'une description textuelle ?
Quelle est la méthode utilisée pour générer la forme 3D colorée ?

À la base de ces problématiques nous avons proposé quatre hypothèses pour atteindre notre objectif clé.

H1: on croit que, le changement de paramètre influe sur le nombre d'epoch et le temps d'entrainement et même pour la qualité des résultats obtenus.

H2: on pense que, l'utilisation de la méthode d'incorporation conjointe (joint embedding) de l'apprentissage par association LBA et apprentissage métrique aide le modèle pour récupérer la forme 3D correspondante à une description textuelle saisie en entrée.

H3: selon les travaux de plusieurs auteurs qui ont fait des approches liées à la génération des textes en images, image en forme et texte en forme on suppose que l'utilisation d'une architecture de réseau génératif adversaire GAN avec n'importe quel type c'est une méthode idéale pour garantir la génération des formes 3D.

H4 : on croit qu'utilisation de l'architecture « CWGAN » aide le modèle pour générer une nouvelle forme 3D qui n'existe pas dans l'ensemble de jeux de données.

Pour effectuer les méthodes de recherche ont été besoin plus d'outils comme l'analyse documentaire (articles …) et aussi on va utiliser deux grandes bases de jeu de données (data Set)

Shape Net et primitive contient un ensemble des formes 3D est augmenté avec des descriptions textuelles.

Par rapport à la structure de notre mémoire, il comporte deux parties principales :

La première partie dite théorique est divisée en deux chapitres :

Chapitre I: dans ce chapitre on va discuter sur l'apprentissage automatique, et de façon plus précise, les réseaux de neurones artificiels et architectures génératives.

Chapitre II: on va présenter les travaux connexes qui sont en relation avec notre approche.

La deuxième partie quant à elle est divisée aussi en deux autres chapitres.

Chapitre III: dans ce chapitre on va montrer les architectures utilisées avec le cadre conceptuel ainsi que les détails de chaque méthode de travail pour arriver à notre objectif.

Chapitre IV: Ici, on va présenter l'implémentation et la discussion des résultats obtenus, et ainsi l'équipement et l'environnement utilisés pour la réalisation de travail.

I. Chapitre

Les concepts de base sur

L'apprentissage automatique et profond

1. Apprentissage automatique (Machine Learning).
2. Apprentissage profond (Deep Learning).

I. Chapitre :Les concepts de base : (apprentissage automatique et profond)

Introduction

Notre premier chapitre comporte la définition des concepts de base de notre thème, il sera scindé d'expliquer les concepts de base des deux parties principales:

La première partie sera consacrée pour définir l'apprentissage automatique ou bien (le machine Learning) et ses trois principales méthodes (supervisées, non supervisé et par renforcement),par rapport à la deuxième partie sera concentré sur l'apprentissage profond (Deep Learning), tout en basant sur certaines architectures utilisées dans la littérature comme les réseaux de neurones artificiels et génératif.

À la fin, on va clôturer notre chapitre par la détermination des métriques qui permet de tester les performances d'un modèle en machine Learning.

1. L'apprentissage automatique (Machine Learning)

1.1. Définition de l'Apprentissage automatique

Les robots **[1]**et les automates sont une source d'intérêt depuis plusieurs siècles. Cependant, le véritable progrès de l'apprentissage automatique n'a commencé que dans les années 1950, à une époque où les ordinateurs n'en étaient qu'à leurs balbutiements et où l'intelligence ne pouvait que vous faire rêver. Au cours des deux derniers siècles, des théoriciens tel que Thomas Bayes, Adrien Marie Legendre et Pierre-Simon Laplace avaient déjà jeté les bases de la recherche, mais ce n'est qu'avec les travaux d'Alan Turing que nous avons réellement parlé d'apprentissage automatique**.** Donc quelle est la définition de l'apprentissage automatique ?
D'après l'auteur Mitchell **[2],** l'apprentissage automatique connu aussi sous l'appellation apprentissage machine, en anglais (Machine Learning), est une technique d'intelligence artificielle permettant à une machine d'apprendre à partir d'exemples déjà disponibles.

- *« On dit qu'un programme informatique tire des leçons de l'expérience E en ce qui concerne une certaine classe de tâches T et une mesure de performance P, si sa performance aux tâches en T, telle que mesurée par P, s'améliore avec l'expérience E. »*

1.2.Les types d'apprentissage

Pour donner à un ordinateur la capacité d'apprendre, on utilise des méthodes d'apprentissage qui sont fortement inspirées de la façon dont nous, les êtres humains, apprenons à faire des choses.

L'une des premières méthodes d'apprentissage **[3]** a été formulée par l'auteur Donald Hebb en 1949, elle était articulée sur l'apprentissage des corrélations. On peut distinguer trois types d'apprentissage :

- **L'apprentissage supervisé** (Supervised Learning).
- **L'apprentissage non supervisé** (Un-Supervised Learning).
- **L'apprentissage par renforcement** (Renforcement Learning).

1.2.1. Apprentissage supervisé

Consiste à définir des règles de comportement à partir d'une base de données contenant des exemples de cas déjà étiquetés. Plus précisément, cette base de données est constituée d'une série de paires entrées sorties $\{(x_i; yi)\}_{1\leq i\leq n}$ aléatoires.

Le but est alors d'apprendre à prédire la sortie **Y** pour chaque nouvelle entrée **X.** C'est ce qu'on appelle la capacité de généralisation. **[4] « Voir là figure 1.1»**

On parle de régression dans le cas où les sorties sont à valeurs continue **[5]**, et de classification dans le cas où elles sont à valeurs discrètes. Donc par exemple en veut apprendre un nouveau langage par exemple le TURC il vous faudra soit acheter un livre de traduction Turque-français, ou bien trouver un professeur de Turque. Le rôle ici du professeur ou du livre de traduction sera de superviser votre apprentissage en vous donne des exemples de traductions français turque que vous devrez mémoriser.

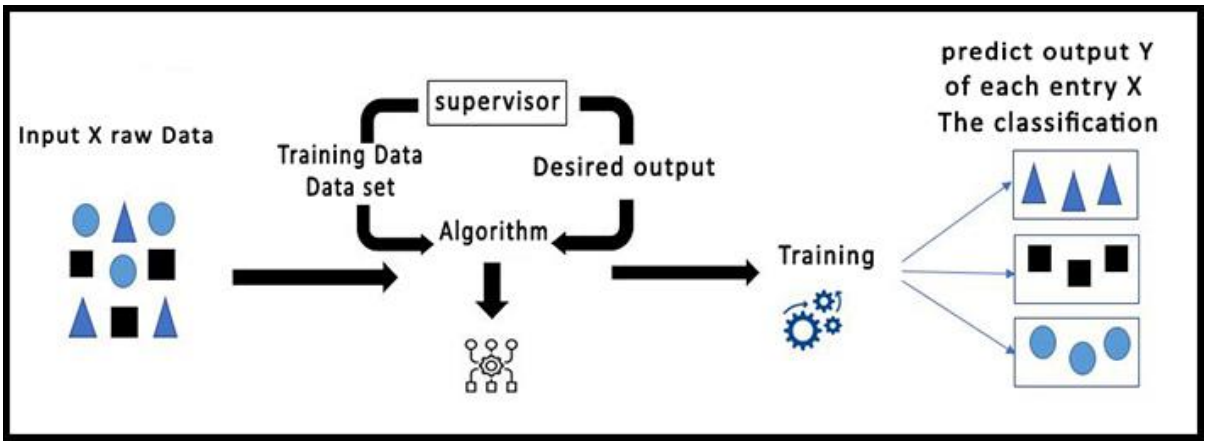


Figure 1- 1:schéma représente l'apprentissage supervise. **[6]**

1.2.2. Apprentissage non supervisé

La théorie **[7]** de l'apprentissage non supervisé, encore appelé apprentissage à partir d'observations ou découverte traite le cas où l'on dispose seulement des entrées. Le problème le plus important consiste alors à effectuer un partitionnement des données, également appelé regroupement. Il s'agit de regrouper les observations en différents groupes homogènes (les clusters), en faisant en sorte que les données de chaque sous-ensemble partagent des caractéristiques communes **« Voir la figure 1.2 »**

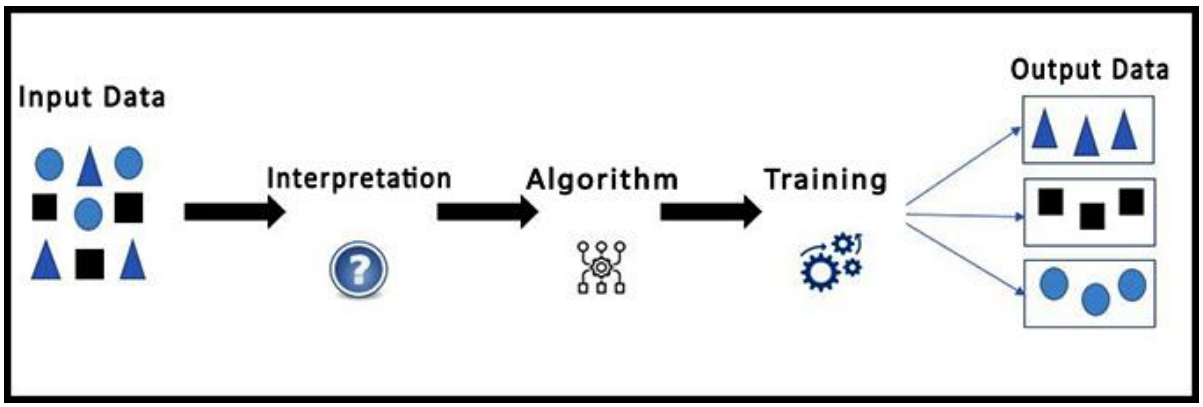


Figure 1- 2:schéma représente l'apprentissage non supervise . **[**8**]**

1.2.3. Apprentissage par renforcement

Les premiers travaux en informatique représentatifs du cadre de l'apprentissage par renforcement (Renforcement Learning) datent approximativement en 1960.**[9]** Cette dernière méthode **[10]** s'inspire de la façon dont nous éduquons nos animaux de compagnie, en leur offrant une friandise quand ils font une bonne action. Cette méthode étant mathématiquement plus avancée que les deux premières. Tel que cet arrière-plan psychologique a été présenté dans une Thèse d'Habilitation à diriger des recherches de l'université Paris VI.

L'apprentissage par renforcement **[11]** est utilisé pour faire apprendre un agent comment se comporter dans un environnement, dans ce cas, aucune donnée (dataset) est disponible, de ce fait, ce type d'apprentissage requis la prise d'un agent et le mettre dans un monde, et une fois cet agent se trouve intégré, il va interagir avec l'environnement et il va construire son propre dataset. A chaque fois l'agent prendre une action il reçoit un état (state) et une récompense (reward) qui peut être positive ou négative **« Voir la figure 1.3 »**

Le but de cet apprentissage est de maximiser le nombre de récompenses.

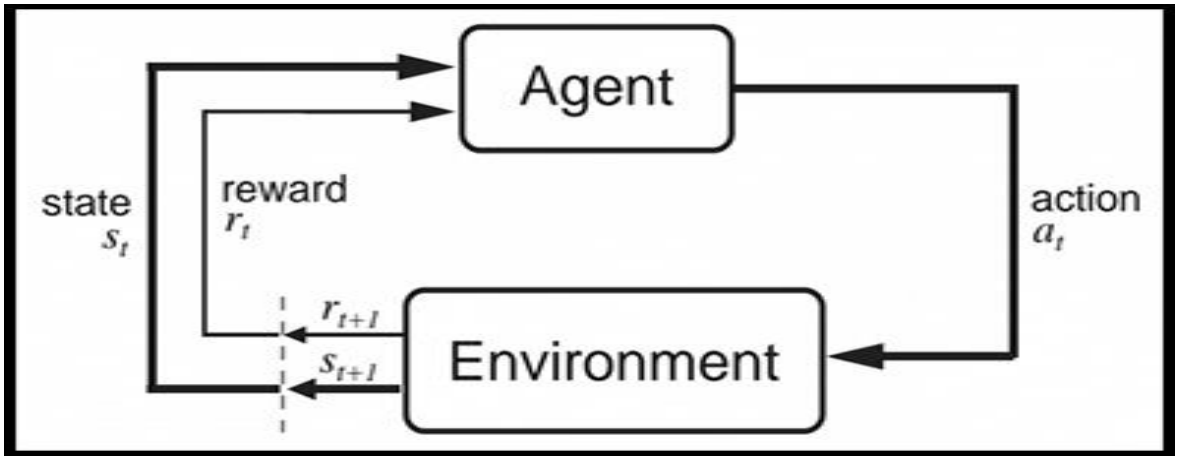


Figure 1- 3:schéma représente l'apprentissage par renforcement . **[12]**

Tel que :

- s_t **:** état perçus au temps t.
- a_t **:** action effectuée au temps t.
- s_{t+1} **:** état perçus au temps $t+1$.
- r_{t+1} : récompense reçus au temps $t+1$.

2. Apprentissage profond (Deep Learning)

L'apprentissage profond (ou Le Deep Learning) a été l'un parmi les plus grands contributeurs à l'avancement de l'IA aussi rapidement qu'aujourd'hui. L'apprentissage profond est une forme (ou un type) d'intelligence artificielle , et un sous-domaine de Machine Learning (ou l'apprentissage automatique) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées. Il utilise un certain type d'algorithme d'apprentissage automatique basé sur des modèles connus sous le nom de réseaux de neurones artificiels (RNA), vaguement inspiré par le cerveau humain.

2.1.Définition des réseaux de neurones artificiels

« *Le concept des réseaux de neurones artificiels* **[13]** *fut inventé en 1943 par deux chercheurs de l'Université de Chicago : le neurophysicien Warren McCullough, et le mathématicien Walter Pitts. Dans un article publié dans le journal Brain Theory, les deux chercheurs présentent leur théorie selon laquelle l'activation de neurones est l'unité de base de l'activité cérébrale.* »

Les réseaux de neurones artificiels **[14]** représentent un système dont sa conception schématique et son fonctionnement est similaire beaucoup plus d'un réseau de neurones biologiques,

ils ont été rapprochés par des méthodes statistique et d'apprentissage de type probabiliste particulier " bayésien". Ensuite, ces réseaux ont été placés d'une part dans la famille des applications statistiques, enrichis avec un ensemble de paradigmes, permettant de créer des classifications rapides (réseaux de Kohonen en particulier), et d'autre part dans la famille des méthodes de l'intelligence artificielle, où les réseaux ont subi un ajout d'informations d'entrée au raisonnement logique formelle (réseaux formels).

A) La fonction de Combinaison et d'activation

Au début, les entrées $(x_1 \dots x_n)$ de chaque neurone sont multipliées avec les poids $(w_1 \dots w_i)$, c'est la somme de produit scalaire entre le vecteur des entrées et le vecteur des poids synaptiques et le terme b, ce qu'on appelle la fonction de combinaison.

$$\boldsymbol{z = b + \sum_i w_i x_i} \qquad (1)\ \textbf{[13]}$$

Tel que :

- Le $\boldsymbol{w_i}$: représenté l'ensemble des poids.
- Le $\boldsymbol{x_i}$: représenté l'ensemble des entrées.
- Le $\boldsymbol{b}$: Le terme b est souvent appelé le biais dans l'équation car il joue le rôle d'un contrôleur dans le neurone est prédisposé à déclencher un 1 ou un 0.
- Pour produire une sortie égale à 1, il faudra une grande valeur en entrée, dans ce cas, on comprend que le biais prend une valeur très élevée.
- Mais, si le cas d'une sortie égale à 0, il permet plus facilement de donner un biais faible.

Il y a plusieurs fonctions d'activation ont été proposées, mais dans notre cas on s'intéresse sur ces trois principales fonctions **:** Sigmoïde et ReLU, soft max**.**

❖ Sigmoïde

La fonction sigmoïde **[15]** considère parmi la plus ancienne et la plus populaire fonction **« voire la figure 1.4».**

Le principe de la fonction de sigmoïde est :

- Le $\boldsymbol{z}$ **:** représente le résultat de la somme pondérée des entrées (**la fonction de combinaison)** .
- Le $\boldsymbol{e}$ **:** est la constante exponentielle, à peu près égale à **2,71828**.

En apparence, l'équation peut sembler compliquée, mais lorsque on comprend comme elle fonctionne, on trouve que c'est une forme très simple à appliquer.

On peut voir que $\boldsymbol{\sigma}(\boldsymbol{z})$ comme une sortie dans le résultat non bornée de **0** à **1** son principe est basé sur :

- Lorsque le $\boldsymbol{z} > 0$ **(***z* positif) : on remplace e^{-z} à **0,** donc $\boldsymbol{\sigma}(\boldsymbol{z})$ se rapproche de 1.
- Lorsque le $\boldsymbol{z} < 0$ ($\boldsymbol{z}$ est nègatif) on remplace e^{-z} par la valeur donnée, dans ce cas le dénominateur croît exponentiellement, et le résultat de $\boldsymbol{\sigma}(\boldsymbol{z})$ se rapproche de **0.**

Lorsque le $\boldsymbol{z} = 0$ ça veut dire il est au centre, Donc le résultat est égal à $\frac{1}{2}$.

$$\sigma_{(0)} = \frac{1}{1+e^{0}} = \frac{1}{2} \qquad (2)\ \textbf{[15]}$$

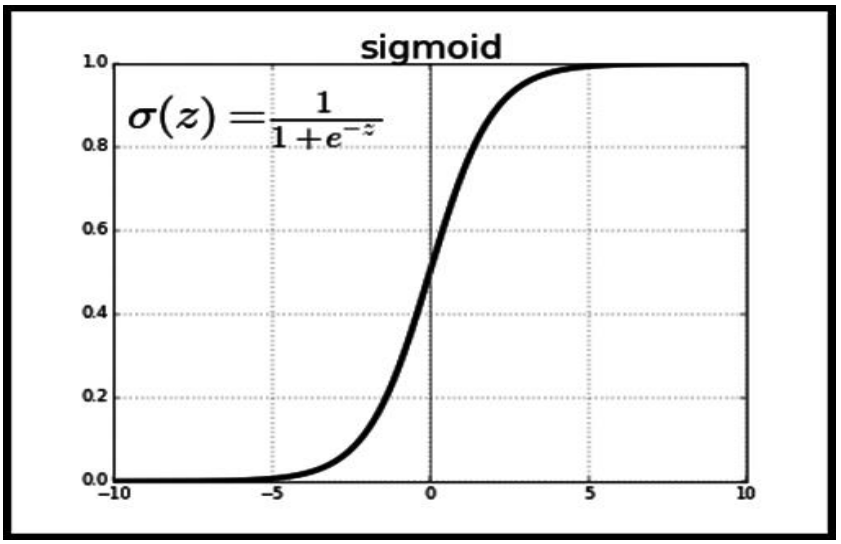


Figure 1- 4:shèma représente la fonction sigmoïde. **[15]**

Comme on a déjà vu précédemment, tous les neurones utilisent la fonction d'activation sigmoïdes mais a force du temps cette dernière est devenue obsolète. Après les chercheurs ont trouvé qu'il existe des neurones qui ont beaucoup de couche à former, en raison du problème du gradient de fuite, pour cela ils ont changé la fonction sigmoïde **[15]** par l'unité linéaire rectifiée, ou ReLU c'est très simple a appliquer.

❖ ReLU

$$R(z) = max(0, z). \quad (3)\ \textbf{[15]}$$

Le principe de ReLU **[15]** est basé sur :

- Dans le cas où les valeurs sont positives, le ReLU laisse ces dernières passer sans changement.
- Dans le cas des valeurs négatives, le ReLU remplace ces dernières simplement par zéro **0**. **« Voir la figure 1.5 ».**

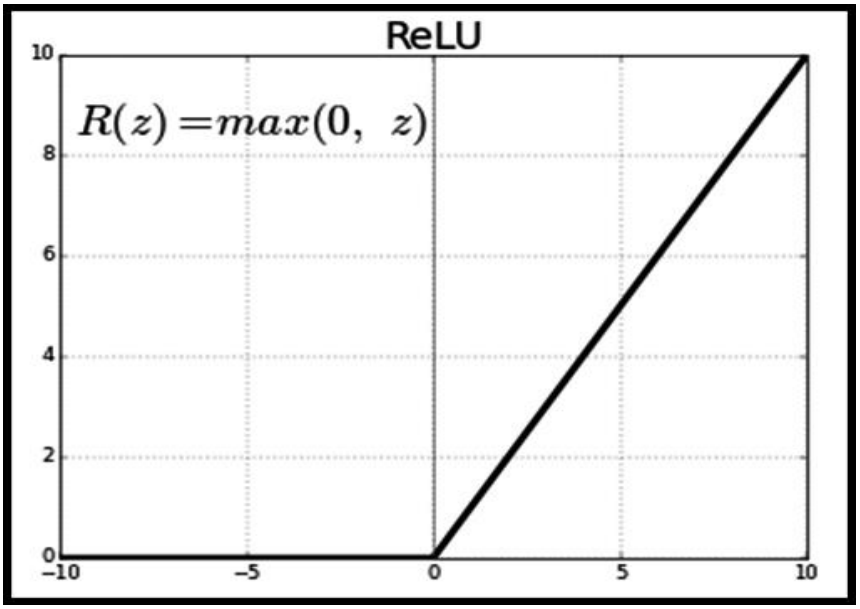


Figure 1- 5:schéma représente la fonction ReLU. **[15]**

La fonction rectifiée Relu avoir d'autre type comme Leaky Relus cette dernière elle est similaires au Relu sauf il y a un petit changement au niveau de calcul par exemple :

- Si le x est supérieur à 0, dans ce cas le Leaky Relu laissent les valeurs passe sans changement.
- Par contre si le x est égal ou inférieur à 0 dans ce cas on dit que l'unité n'est pas active, donc le Leaky Relu permettent de multiplier les valeurs de x avec un gradient positif égale à 0.01.

$$f(x) = \begin{cases} x & si\ \ x > 0\,, \\ 0{,}01x & sion \end{cases} \quad (4.1)\ \textbf{[15]}$$

La fonction sigmoïde est une fonction utile pour les classifications binaires, mais dans le cas où il y a la multi classification ! Quelle sera la fonction utilisée ?

Dans ce cas, il faut définir un autre type de fonction, est appelé le soft max.

❖ Soft Max

La régression Soft max **[15]** est un autre type de la fonction d'activation et aussi est une forme de régression logistique, cette dernière est utile dans le cas où on veut d'appliquer une classification non binaire (la multi classification).

Le Soft max permet d'attribuer des probabilités aux valeurs d'entrée, le total de ces probabilités doit être égale à 1. Les valeurs de sortie se situent dans la plage [0,1] cette fonction permet d'améliorer la rapidité d'apprentissage.

Le soft max **[15]** est appliqué dans la couche avant dernière de la couche résultat, et aussi nous n'oublions pas que les deux couches doivent d'être comportées le même nombre des nœuds.

D'une manière générale, le principe de soft max **[16]**permet de calculer les probabilités d'une classe sur toutes les classes possibles qu'on a. Cette méthode permet de définir les probabilités de chaque classe, selon les résultats obtenus, la probabilité la plus élevée représente automatiquement la classe cible pour les entrées données. **« Voir la figure 1.6».**

La fonction soft max est représenté sous la forme :

- e^{xj}: représente l'exponentielle de la valeur d'enter.
- $\Sigma_i e^{xi}$**:** représente la somme de toutes les valeurs des entrées.

Le rapport de e^{xj} sur la somme des e^{xi}représente la sortie de soft max. lorsque le soft max calcule la probabilité pour chaque classe possible, on dit qu'on est dans le soft max complet.

$$\sigma(x_j) = \frac{e^{xj}}{\Sigma_i e^{xi}} \quad (4)\ \textbf{[15]}$$

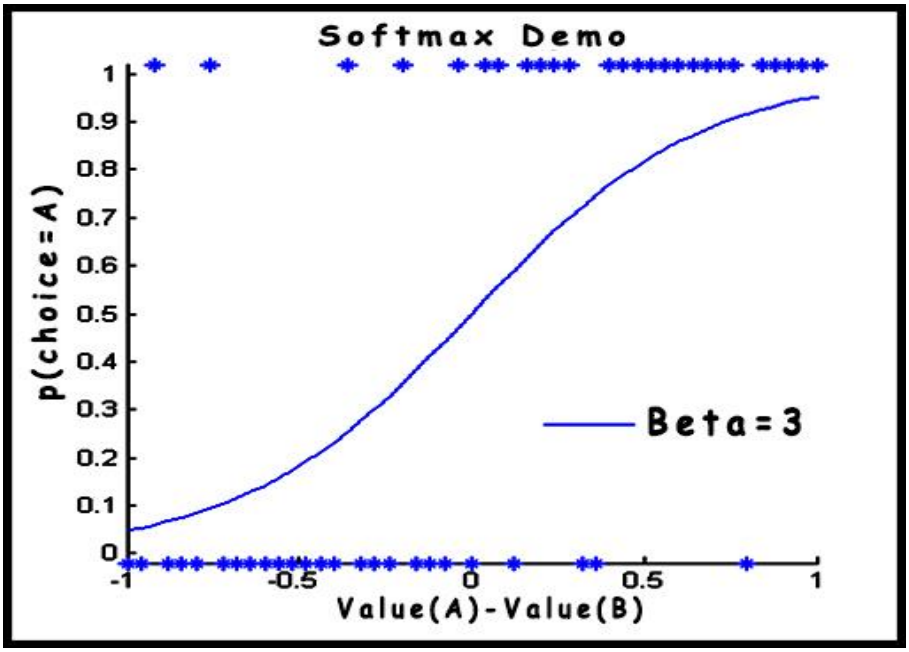


Figure 1- 6:schéma représente la fonction soft max. **[17]**

B) Algorithme d'entrainement

L 'apprentissage **[18]** est une phase du développement d'un réseau de neurones, durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré.

Tel que tous les réseaux de neurones possèdent un algorithme d'entraînement pour mettre à jour les poids synaptiques des données présentées en entrée du réseau, le but de cette dernière est d'entraîner le réseau de neurones « d'apprendre » à partir des exemples (un jeu de donnée).

La première étape consiste à définir ce qu'on entend par entraîner un réseau sur une base de données (training set), et ce, pour qu'il puisse associer la sortie désirée à une entrée que l'on appelle l'apprentissage supervisé.

Dans la suite, On note $\boldsymbol{E}$ la liste de tous les paramètre du modèle $\boldsymbol{E} = \{\boldsymbol{wi}, \boldsymbol{bi}\}$.

Pour mesurer les performances d'un ensemble de paramètres, on utilise une fonction de perte (Loss function) qui calcule l'erreur de prédiction sur l'ensemble des éléments de la base de données. Le but de l'apprentissage est de minimiser la valeur de cette fonction en faisant varier les valeurs de l'ensemble $\boldsymbol{E}$.

Un exemple de la fonction d'erreur est l'erreur quadratique Moyenne (mean squared error {MSE}).(Equation 6)

$$E = \frac{1}{2}\sum_{k \in K}(Y_k - t_k)^2 \qquad (6)$$

- **E** représente la fonction d'erreur.
- Y_κ Représente la prévision (sortie du réseau).
- t_k Représente de la valeur cible (valeur attendue) pour la *ième* observation.

Dans le cas où la sortie de réseau fourni une réponse très proche des valeurs de jeu de données, on dit que l'entrainement de réseau a été bien réalisée, sinon on passe à l'algorithme de rétropropagation.

2.2.Les types de réseaux de neurones artificiels

2.2.1. Réseau de neurone Convolutionnel CNN

Le réseau de neurones convolutionnel **[19]** désigné par l'acronyme CNN, en anglais (Convolutional Neural Networks) ,il est considéré l'un parmi les meilleurs réseaux neurones, et les plus performants modèles de classification, de reconnaissance des (images / vidéo) et de traitement automatique du langage naturel (TALN).

❖ Les étapes de fonctionnements de CNN pour la classification des images :

En entrée, une image est représentée sous forme d'une matrice de pixel en 2 dimensions de niveau de gris et les couleurs fondamentales RVB [Rouge, Vert, Bleu] en anglais RGB [Red, green, Blue]. Les images sont représentées par une troisième dimension, de profondeur 3

Dans la première partie, le CNN est considéré comme extracteur de caractéristiques des images à travers les passages d'une image dans les successions de filtres, puis le noyau de convolution crée de nouvelles images appelées cartes de convolutions.

Après, une opération de maximum local sera appliquée sur ces cartes convolutions, et ce, en vue de réduire la résolution de l'image. À la fin, les nouvelles cartes seront transformées en vecteur 2D puis en vecteur 1D appelé code CNN.

Ce code CNN se trouve entre la sortie de la première partie convolutive et l'entrée de la deuxième partie perceptron multicouche.

Le rôle de cette dernière est la classification des images par les caractéristiques du code CNN combiné. Concernant la dernière couche, elle comporte un neurone par catégorie pour classifier les images, elle est normalisée entre des valeurs numériques entre 0 et 1 pour la production de la distribution de probabilité sur les différentes catégories. **« Voir la figure 1.7 »**

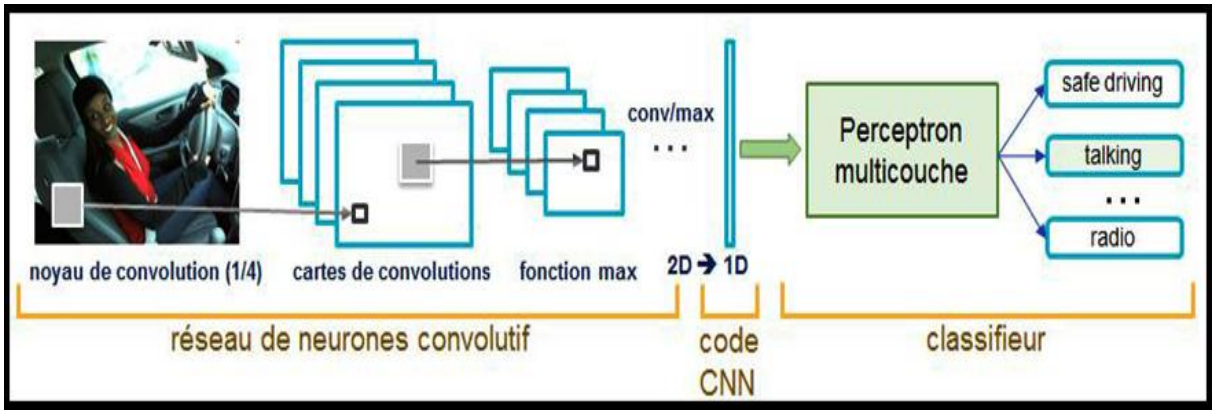


*Figure 1- 7:Architecture standard d'un réseau de neurone convolutionnel.***[20]**

A) Les Couches de CNN

❖ La couche de convolution

La couche convolutionnelle est un composant important dans le réseau CNN, tel que les paramètres de la couche consistent en un ensemble de filtre ou noyaux .

Trois hyperparamètres, appelés aussi le volume de sortie, permettent de dimensionner le volume d'une couche convolutionnelle, tel que :

- **La profondeur** : représente le nombre de noyaux /neurones .
- **Le pas** :contrôle le chevauchement des champs récepteurs, tel que plus que le pas est petit ,plus les champs récepteurs se chevauchent et plus le volume de sortie sera plus grand .

La marge a 0 ou (0 padding) :cette marge permet de contrôler la dimension spatiale du volume de sortie, il est de préférence de conserver la même surface que celle du volume d'entrée.

La première couche de réseau de neurone est généralement considérée comme la plus importante couche de CNN, Son but est de déterminer la présence d'un ensemble de caractéristiques dans les images reçues en entrée. Dans ce cas, on utilise la méthode de filtrage par convolution :

1."Glisser" la fenêtre qui représentant la feature sur image(kernel).

2. "Calculer "le produit de convolution entre la feature (filtre ou kernel) et chaque portion de l'image balayée.

En entrée , la couche reçoit une ou plusieurs images, chaque image sera transformer en matrice de pixel, la matrice sera devisée en sous parties ayant la même taille que la matrice de filtre(Karnel), après, chaque sous partie seront multipliée par le filtre, tout en respectant les chauvechements de filtrage.

La matrice de filtre rassemble exactement les caractéristiques que l'on souhaite retrouver dans les images .A la fin de cette opération on obtient les cartes d'activation (feature map) de chaque paire (image, filtre), ces dernières sont celles qui nous précisent où se situent les caractéristiques (les features), en notant que plus la valeur est élevée, plus l'endroit convient à une partie de l'image . **« voir la Figure 1.8 »**

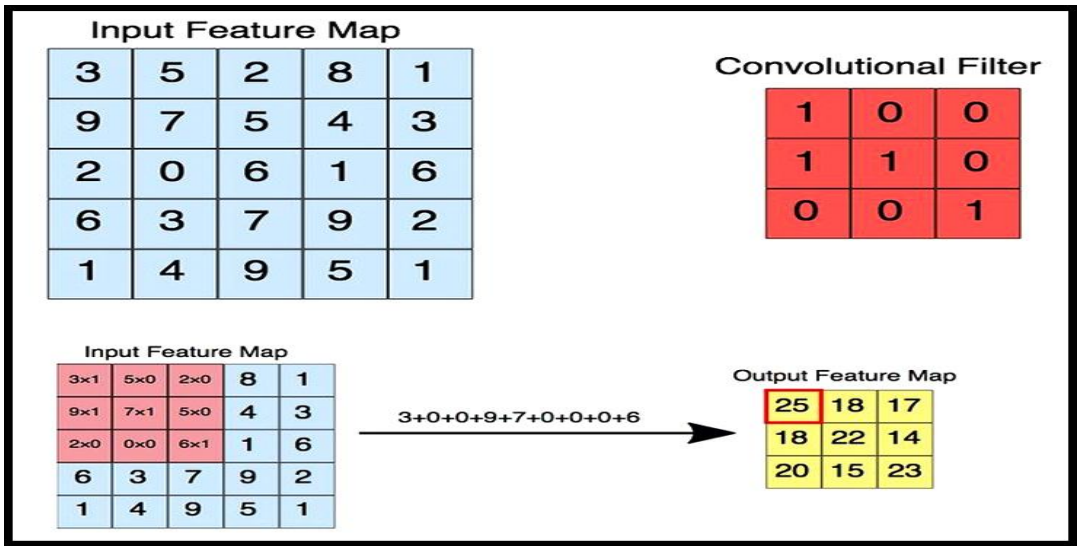


Figure 1- 8:schéma représente les convolutions d'une image avec le filtre.**[21]**

❖ **Convolution3D**

La couche convolution 3D est un type de la couche convolution standard, tel que les convolutions 3D **[22]** appliquent un filtre tridimensionnel sur l'ensemble de données, le filtre se déplace dans 3 directions (x, y, z) pour calculer les représentations d'entités de bas niveau, en effet, le format de sortie de chaque filtre, devient un espace de volume tridimensionnel (cube). Ces filtres 3D (cubes) sont utiles pour la détection d'événements dans les vidéos, les images médicales 3D, etc. Ils ne sont pas limités à l'espace 3D mais peuvent également être appliqués aux entrées d'espace 2D telles que les images. La convolution 3D est obtenue en convoluant un noyau 3D au cube formé en empilant plusieurs cadres contigus ensemble. Par cette construction, les cartes d'entités dans la couche de convolution sont connectées à plusieurs trames contiguës dans la couche précédente. Formellement, la valeur à la position (x, y, z) sur la jème carte d'entités dans la ième couche est donnée par , **« voir la figure 1.9 ».**

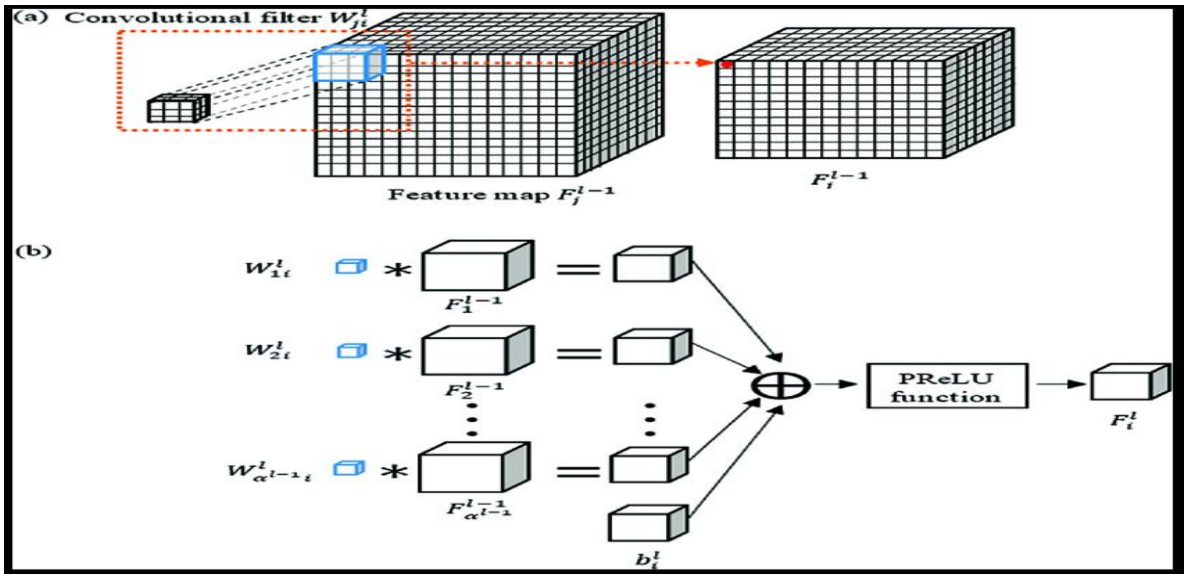


Figure 1- 9:schéma représente 3D- convolution.**[23]**

❖ **La couche de ReLU (Unités linéaires rectifiées)**

Après le filtrage des images par la couche convolutionelle, on passe à la deuxième couche de CNN est la couche Relu, cette dernière se charge à rectifier les valeurs de chaque carte de convolution obtenues d'après le filtrage.

La fonction d'activation ReLU est certainement la plus utilisée dans les CNN profonds **[24]** car elle permet : une optimisation plus facile, de fournir des réponses claires, et de réduire les problèmes de gradient de fuite. En effet, le sigmoïde et la tangente hyperbolique ont l'inconvénient de renvoyer de très faibles gradients lorsque la valeur absolue de l'entrée est grande.

Le réseau éprouve alors des difficultés à mettre à jour ses paramètres lors de la phase d'apprentissage. La fonction ReLU renvoie un gradient constant pour une entrée importante, permettant ainsi d'apprendre plus rapidement (en particulier les réseaux d'une certaine profondeur). Donc, la couche de correction ReLU (Rectified Linear Units) joue le rôle d'une fonction d'activation comme on a vu dans la partie précédente (la fonction d'activation Relu), elle permet de corriger toute les valeurs négatives reçues en entrées et les remplacer par le zéro, la fonction est réelle non-linéaire définie par **ReLU(x)=max(0,x)**,**« voir la figure 1-10 ».**

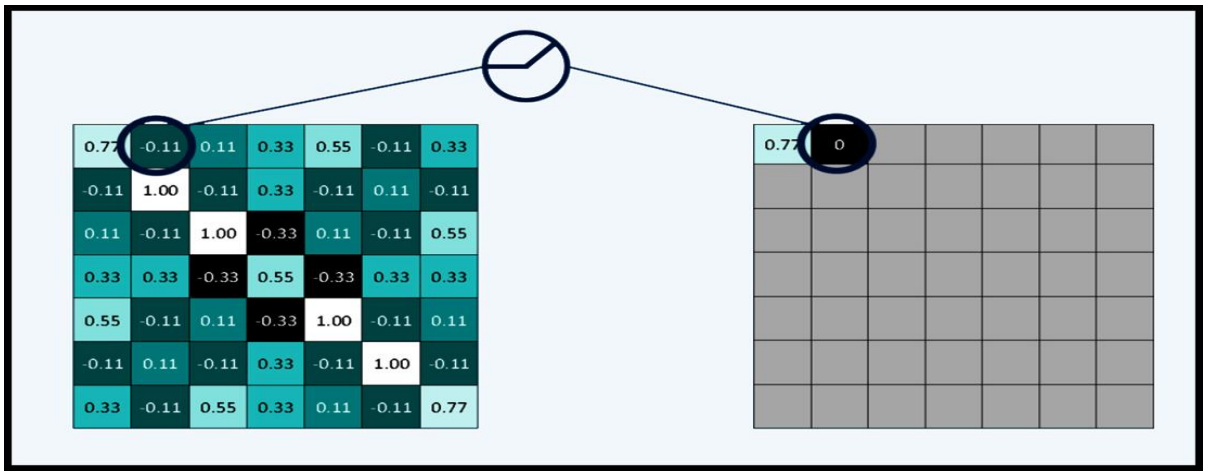


Figure 1- 10:schéma représente la couche ReLU. **[25]**

❖ **La couche de pooling**

La couche de pooling est la troisième couche de CNN, son principe permet de prendre en considération les valeurs positives rectifiées de chaque carte de convolution (images) d'après la couche précédente (couche Relu). Son objectif consiste à réduire la taille des images avec la préservation de leurs caractéristiques importantes, les types de pooling les plus utilisés sont le « Max pooling » et « l'average pooling ».

❖ **Max pooling**

L'objectif de Max pooling consiste à découper l'image en cellules régulières carrées en petite taille, et ce, pour ne pas perdre beaucoup d'information toute en sauvegardant la valeur maximale dans chaque cellule.

Généralement le choix de la taille d'une cellule est compris entre 2x2 pixels pour éviter les chevauchement, ce choix nous permettre d'obtenir le même nombre de feature maps en sortie qu'en entrée, mais avec une petite taille.

Cette réduction de taille et ce nombre réduit de paramètre permet d'améliorer l'efficacité du réseau en évitant le sur-apprentissage. **« voir la Figure 1.11 »**

Exemple : *« lorsqu'on veut reconnaître un chien par exemple, ses oreilles n'ont pas besoin d'être localisées le plus précisément possible : savoir qu'elles se situent à peu près à côté de la tête suffit ! »*

La pooling ne s'intéresse pas beaucoup par la position exacte des caractéristiques (features), si le feature se situe un peu plus haut ou un peu plus bas, et même s'il y aura une légère orientation différente, elle ne provoquera pas un changement radical dans la classification de l'image.

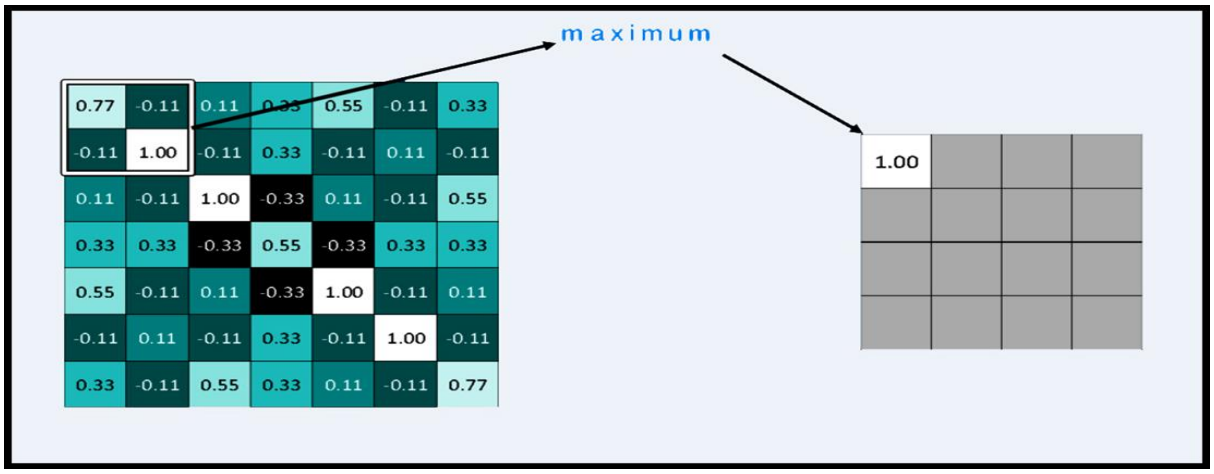


*Figure 1- 11:schéma représente la couche pooling.***[26]**

❖ **Average pooling**

L'average pooling a le même principe que le Max pooling, tel qu'il permet de renvoyer la moyenne des éléments sur une fenêtre de calcul.

❖ **La couche fully-connected**

La couche fully-connected est la dernière couche du réseau de neurone convolutif. Son fonctionnement est basé sur deux vecteurs, l'un est présenté en entrée et l'autre est produit en sortie. Pour cette opération le fully-connected applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

La dernière couche fully-connected est spécialisée pour la classification des images, tel que, dans la phase d'entrée, elle renvoie un vecteur de taille N (ou N le nombre de classes présenté dans la classification des images).

Le vecteur possède un ensemble d'éléments, chacun d'eux indiquent la probabilité d'une l'image en entrée d'être appartenu à une classe.

A titre d'exemple, si le problème consiste à distinguer les chats et les chiens, le vecteur final sera de taille 2. Cela veut dire que le premier élément de vecteur représente la probabilité de la classe « chat » et le deuxième élément représente la probabilité de la classe « chien». Si on a dans ce cas [0.9 , 0.1], cette valeur signifie que l'image à 90% de chances de représenter un chat.

Le calcul de la probabilité dans la couche fully-connected est basée sur le produit scalaire de chaque élément de vecteur en entrée par la matrice de poids (fonction de combinaison), puis on applique la fonction d'activation :

Une fois les valeurs en entrée et en sortie soient connectées totalement, la couche fully-connected commence à présenter le résultat. **« voir la figure 1.12»**

à chaque fois on obtient une valeur élevée **[27]** porche de 1 (selon la couche pooling), il devient claire que cette dernière indique la localisation de feature dans l'image, en effet, on peut déterminer à quelle classe appartient cette image .

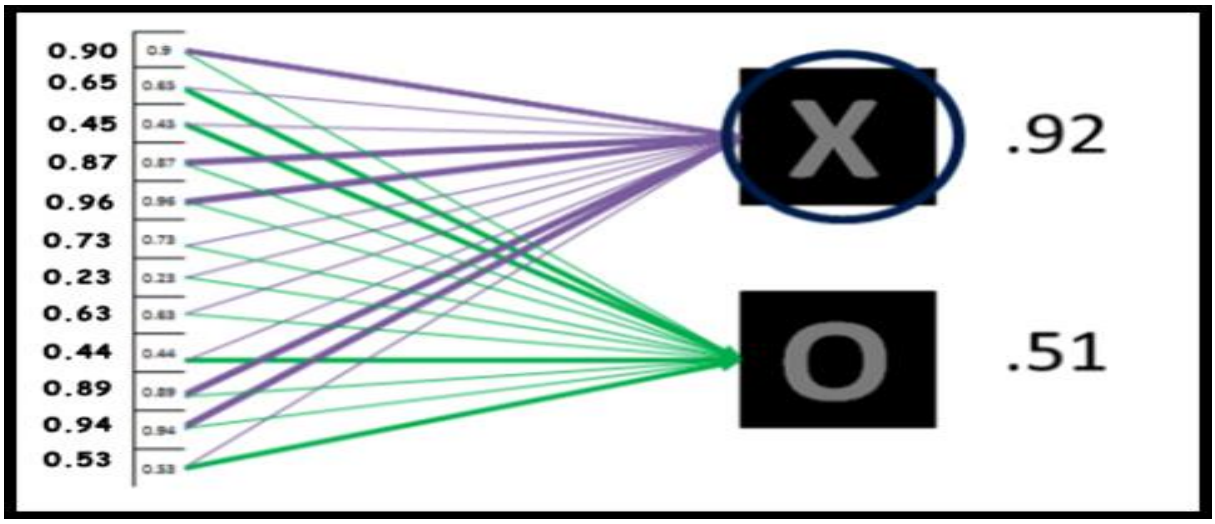


Figure 1- 12:schéma représente la couche fully-Connected **.[26]**

❖ **La batch normalisation**

Cette technique a été [**28]** introduite par l'auteur Sergey Ioffe **[29]** afin d'apprendre les CNN de manière plus rapide et efficace. Elle part de l'observation que pendant l'apprentissage, la distribution des entrées des différentes couches du réseau se change à chaque itération.

Cela induit une adaptation permanente des paramètres du CNN à ces différentes distributions, ce qui augmente le temps d'apprentissage. L'idée de la batch normalisation consiste à normaliser les entrées de chaque couche afin que les distributions de celles-ci soient de moyenne nulle et de variance unitaire.

❖ **Fonction d'activation de la dernière couche**

La fonction d'activation appliquée à la dernière couche entièrement connectée est généralement différente des autres fonctions, pour qu'elle soit appropriée, elle doit être sélectionnée pour chaque tâche. Dans la tâche de classification multi-classe, on choisit le soft max comme fonc-

tion d'activation, du fait qu'elle normalise la valeur de sortie réelle de la dernière couche entièrement connectée à la probabilité de classe cible, où chaque valeur est comprise entre 0 et 1, et toutes les valeurs sont résumées à 1.

« voir la figure 1.13»

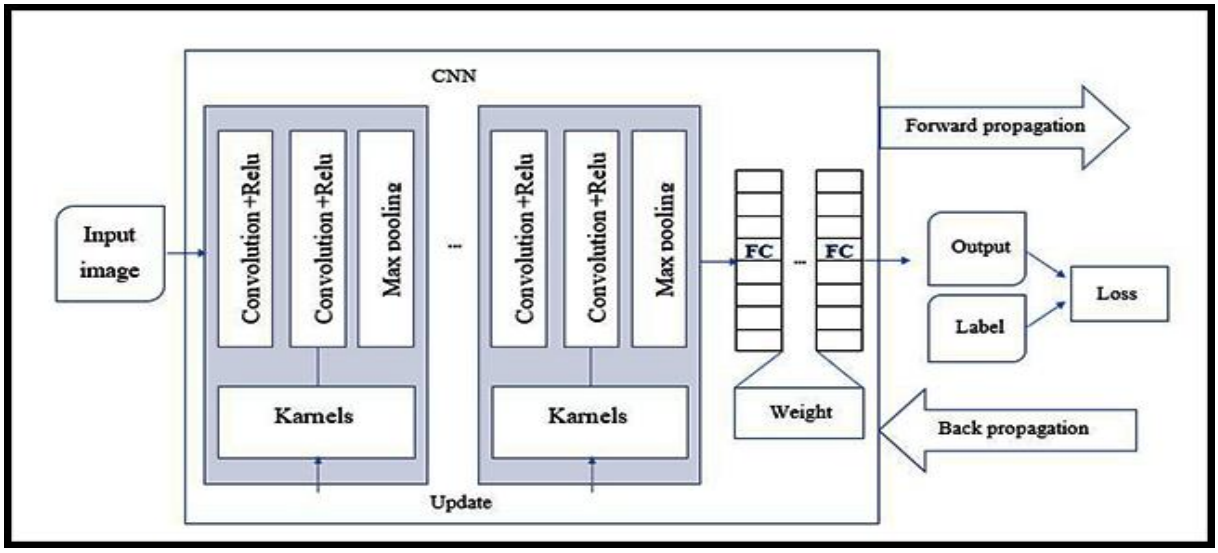


Figure 1- 13:schéma représente les couches de CNN.**[30]**

2.2.2. Réseau de neurone récurrent RNN

Le réseau de neurones récurrent est désigné par l'acronyme RNN en anglais (récurrent neural network), il représente un type de réseau de neurones artificiels conçus pour le traitement automatique du langage naturel (TALN), il se considère ainsi comme un type puissant dans la prédiction des résultats selon le contexte de scénarios.

Le RNN est composé de trois couches à savoirs : la couche d'entrée, la couche intermédiaire (dite couche cachée) et la couche de sortie.

Selon la figure ci-dessous, il apparait clairement que les valeurs de la couche d'entrées sont reliées à la totalité des neurones de la couche adjacente, et que le principe de RNN traite l'information en deux sens, c'est pour cela qu'on l'appelle " un neurone récurrent " .

Pour donner le résultat de l'entrée vers la sortie, le neurone d'une couche particulière **[31]**fait une appelle récurrente d'un neurone précédent grâce aux **« boucles de rétroaction »** , par contre le CNN traite les informations dans un seul sens de l'entrée vers la sortie. Dits « feed forward ». **« Voir la figure 1.14 »**

Le but d'utilisation des boucles de rétroaction dans le RNN, est de traiter les séquences de données, donc on peut dire que ces boucles fonctionnent comme une mémoire. Comme on a déjà indiquer au début, les RNN sont dédiés pour le traitement de langage naturel, de ce fait, ils sont généralement utilisés dans les modèles linguistiques, son but principal consiste à prédire la prochaine lettre dans un mot ou bien le prochain mot dans une phrase d'après les données précédentes.

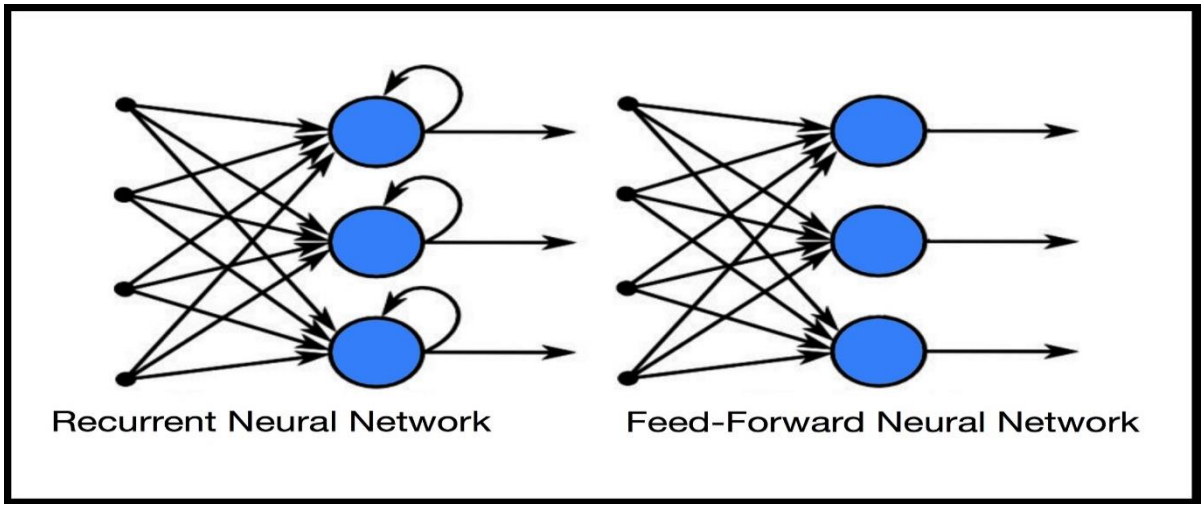


Figure 1- 14:Architecture standard d'un réseau de neurones récurrent **.[32]**

A) Model LSTM

Long Terme Short Terme Memory Networks, désigné par l'acronyme LSTM est un modèle prend son origine de RNN, il a été proposé par les auteurs Sepp Hocheriez et Jürgen Schmid Huber en 1997.

Le LSTM est un modèle de type feedback, son architecture permet d'accumuler les informations en cours de fonctionnement, elle permet aussi d'utiliser les retours récurrents des éléments précédents pour trouver des informations.

La base de l'avènement des LSTM est qu'il existe une ressemblance entre les FNN (feed forward neural network) et les RNN, sauf que le premier (FNN) est un réseau de neurale classique avec un seul appel.

Les FNN proposent des mécanismes qui sont applicables au RNN, mais le problème est que les FNN se caractérisent par un certain nombre d'inconvénients, au fait qu'ils utilisent le principe de la réduction de leur capacité pour résoudre les problèmes plus complexes. En conséquence, la communauté a remis en cause ces inconvénients, en profitant de l'apparition de LSTM basé

sur le principe de long terme et court terme mémoire pour résoudre le problème, tout en se basant sur deux de ces principes :

- LSTM avoir des unités qui sont utilisées pour la construction de la couche RNN.
- LSTM permettent aux RNN de souvenir de leurs intrants à longue période, parce que ce dernier garde les informations dans sa mémoire, et ce grâce à la ressemblance de son principe de conservation d'information avec celui d'un ordinateur au niveau des opérations d'écriture et de suppression d'informations.

La mémoire de LSTM vue comme une cellule porte (gated), ça veut dire elle décide de stocker ou de supprimer des informations selon leurs importances.

L'importance est prise sur la base des poids appris par un algorithme, cela veut dire qu'avec le temps, la porte apprend quelle information est importante par rapport à les autres.

LSTM se compose en trois portes principales **[33] :** entrée, oublier et sortie porte.

- **la porte d'entrée** :détermine quelle information entre.
- **la porte oublier:** supprime l'information n'est pas importante.
- **la porte sortie:** laisser influencer la sortie au pas de temps courant .

« Voir la figure 1.15 » .

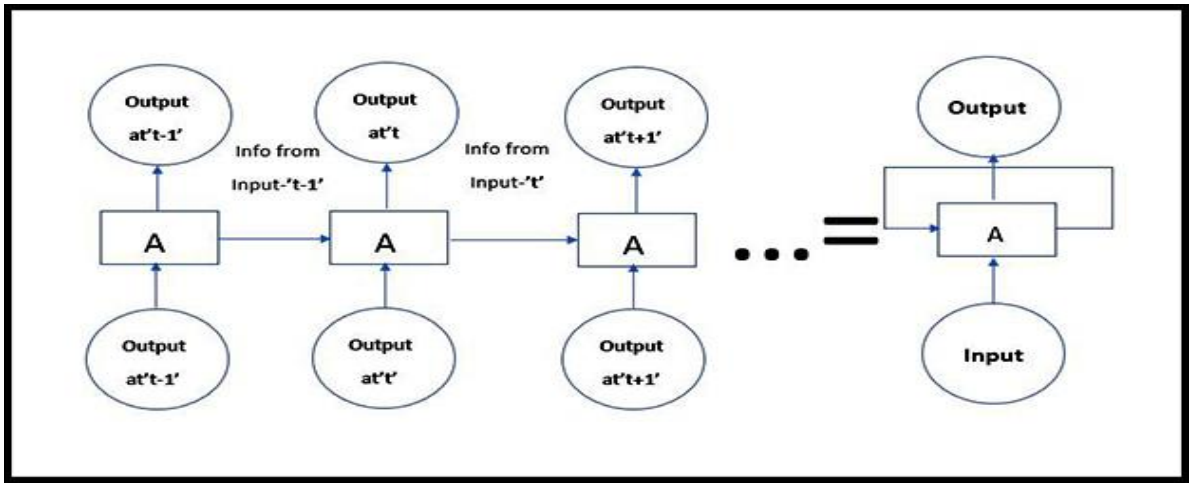


*Figure 1- 15:Architecture standard de LSTM.***[34]**

B) Model GRU

Les Gated Récurrents Unit désignés par l'acronyme (GRU) sont des modèles de RNN qui ont été créés par l'auteur Cho **,** son fonctionnent est similaire de celui de LSTM.

Les études de l'auteur Chung montrent que l'utilisation de GRU, LSTM et RNN classiques est requis pour le traitement de langage naturel (TALN).

D'après l'auteur Greff **[35]**, les GRU sont privilégiés en termes de la puissance de calcul que LSTM, mais les deux modèles restent toujours plus performants que les RNN classique.

Les GRU comprennent deux passerelles, une de réinitialisation, et l'autre de mise à jour.

- **La passerelle de réinitialisation :**détermine la façon de fusionnement de la nouvelle entrée avec la mémoire précédente.
- **La passerelle de mise à jour:** spécifie la quantité de mémoire précédente à conserver.

En comparant le modèle de GRU avec le modèle LSTM **[36]**, on distingue quatre différences principales **:**

- les GRU s'entrainent plus rapidement qu'un LSTM .
- les GRU fonctionnent mieux avec moins de données de formation, sont très simple et faciles à modifier qu'un LSTM.
- Les GRU ont également une complexité de paramètre inférieure à celle de LSTM.
- Les RNN classique fonctionnent sans porte de contrôle de flux d'information entre les cellules. **« Voir la figure 1.16 »**

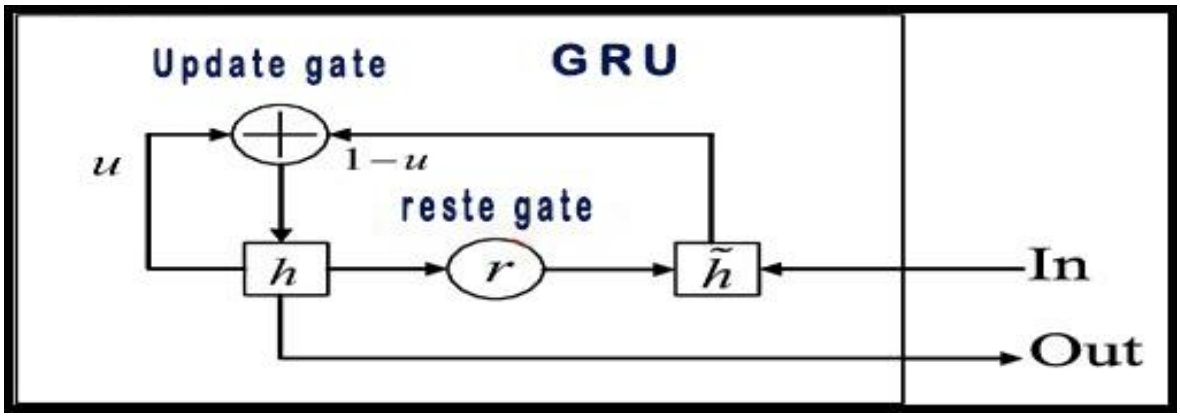


*Figure 1- 16:Architecture standard de GRU.***[37]**

2.3.Réseau génératif

Les réseaux génératifs sont basés sur le principe de générateur et discriminateur pour le traitement (des images ,de texte et de sons ...), il est composé de différents types tel que : GAN,CGAN,WGAN.

A) Générative adversaire networks GAN

« Les réseaux antagonistes génératifs ou GANs **[38]** *(Générative Adversaire Network), sont des algorithmes d'apprentissage non à base de réseaux de neurones artificiels, qui permettent de modéliser et d'imiter n'importe quelle distribution de données. Ils peuvent être utilisés dans différents domaines (traitement d'images, de texte, de sons, ...). Depuis leur*

invention en 2014, les GANs ont suscité un grand intérêt et plusieurs chercheurs ont souligné leur potentiel. »

A titre d'exemple, imaginons que l'on possède une base de données composée de peintures de différentes époques, parmi lesquels des tableaux de Jérôme Bosch, on pourrait entraîner un algorithme de classification pour déterminer si un nouveau tableau a été peint par Bosch ou non, on pourrait également entraîner un algorithme génératif pour créer de nouveaux tableaux (fake)dans le style de Bosch.

Concrètement, l'architecture d'un GAN est composée de deux réseaux de neurones, mis en compétition. Le premier, appelé générateur, il est destiné à créer un échantillon de données. D'après l'exemple précédent, ce serait un faux tableau avec le style de Bosch.

Le deuxième réseau, appelé discriminateur ou (l'adversaire de générateur), il tente de détecter si cet échantillon est original ou bien il s'agit d'une création de son adversaire (le générateur). Ainsi dans notre exemple, le discriminateur essayerait de détecter si une image est une peinture réalisée par Bosch où il s'agit d'une peinture produite par le générateur.

Donc on constate que l'adversaire(discriminateur) n'est pas vraiment un adversaire, cela apparaît comme un enseignant ou un critique bienveillant.

Quoi qu'il en soit, le générateur et l'adversaire forment ensemble un réseau générateur adversarial ou GAN, on note que ces deux deniers (générateur et l'adversaire) ont été inventés en 2014 par l'équipe de recherche du Canadien Ian Goodfellow **[39]**, tel qu'il a inspirée l'Approche des GAN de la théorie des jeux : jeu minimax à 2 joueurs (équation .7)

$$\min_G \max_D V\ (D;G) = \ E_{x\sim pdata}[\log D(x)] + \ E_{z\sim p_z(z)}[\log(1 - D\big(G(Z)\big))] \qquad (7)$$

- **D** représente le discriminateur.
- **G** représente le générateur.
- $\boldsymbol{D(x)}$ C'est la probabilité que les données réelles $\boldsymbol{X}$ soient réelles
- $\boldsymbol{D\big(G(Z)\big)}$ C'est la probabilité que les fausses données($\boldsymbol{G(Z)}$) sont réelles.
- $\boldsymbol{E_{x\sim pdata}}$ C'est la probabilité attendue sur toutes les instances de données réelles $\boldsymbol{x}$
- $\boldsymbol{E_{z\sim p_z(z)}}$ C'est la probabilité attendue sur toutes les instances de fausse donnée$\boldsymbol{z}$

L'objectif du discriminateur est de maximiser $V\ (D\ ;G)$ Ceci est réalisé lorsque

$$D\ (x)\ =\ 1 \text{ Et } D\ (z)\ =\ 0, \text{ où } V\ (D,G)\ =\ 0 \qquad (8)$$

Et le générateur veut minimiser $V\ (D, G)$, donc il veut que $D\ (x)$ et $D\ (z)$ soient aussi petits que possible.

❖ **L'effondrement du mode (mode collaps)**

Habituellement, il est préférable que le GAN produise une grande variété de sorties, c'est à dire une sortie différente (aléatoire) pour chaque entrée de votre générateur.

Alors, l'effondrement du mode **[40]** se produit lorsque le générateur génère une diversité limitée d'échantillons, ou même le même échantillon, quelle que soit l'entrée.

Si le générateur commence à produire la même sortie (ou un petit ensemble de sorties) encore et encore, la meilleure stratégie du discriminateur est d'apprendre à toujours rejeter cette sortie.

Mais si la prochaine génération de discriminateur est bloquée dans un minimum local et ne trouve pas la meilleure stratégie, il est trop facile pour la prochaine itération du générateur de trouver la sortie la plus plausible pour le discriminateur actuel. Donc le générateur apprend à tromper **[41]** le discriminateur en ne générant qu'une variété limitée de données.

B) Conditionnel GAN (CGAN)

Le CGAN est une extension du GAN, l'idée a été d'abord publiée dans un article intitulé « Pluralisme de la police conditionnelle » par les auteurs Mehdi Mirza et Simon Osenero **.** Le CGAN est utilisé comme cadre d'apprentissage automatique pour des modèles de formation composés.

Comme on a déjà mentionné que les GAN sont composées de deux réseaux : le générateur G(z) et le discriminateur D(x), son rôle est de cherche à produire des images quelconques qui se ressemblent à celles d'un jeu de données, le problème est que ce dernier est difficile pour générer et spécifier les caractéristiques souhaitées exactement. Pour cette raison ils ont développé les CGAN basées sur la méthode d'apprentissage en profondeur, ce dernier s'intéresse à conditionner les deux réseaux le générateur et le discriminateur par un paramètre conditionnel, tel que les étiquettes de classe ou bien les données d'autre modalités.

Cette façon permettre d'ajouter une certaine valeur pour le modèle, en vue qu'il puisse apprendre la cartographie multimodale des entrées aux sorties, et ce, par l'alimentation avec plus d'information de différents contextes.

Dans ce contexte, les deux auteurs Mirza et Osiander ont parlé de deux expériences :

La première est unimodale, consiste à former un CGAN sur les images MNIST à 784 dimensions, ce dernier été conditionné par leurs étiquettes de classe. Dans ce cas on peut remarquer que les résultats génères sont meilleurs que les résultats de GAN non conditionnel.
La deuxième est multimodale, elle permet de générer des distributions en fonction des caractéristiques de l'image.

D'après ces deux expériences **[42]** on constate que le CGAN est l'un parmi les réseaux génératifs les plus importants, du fait qu'il accepte plus d'explorations et utilisations. **« Voir la figure 1.17 »**

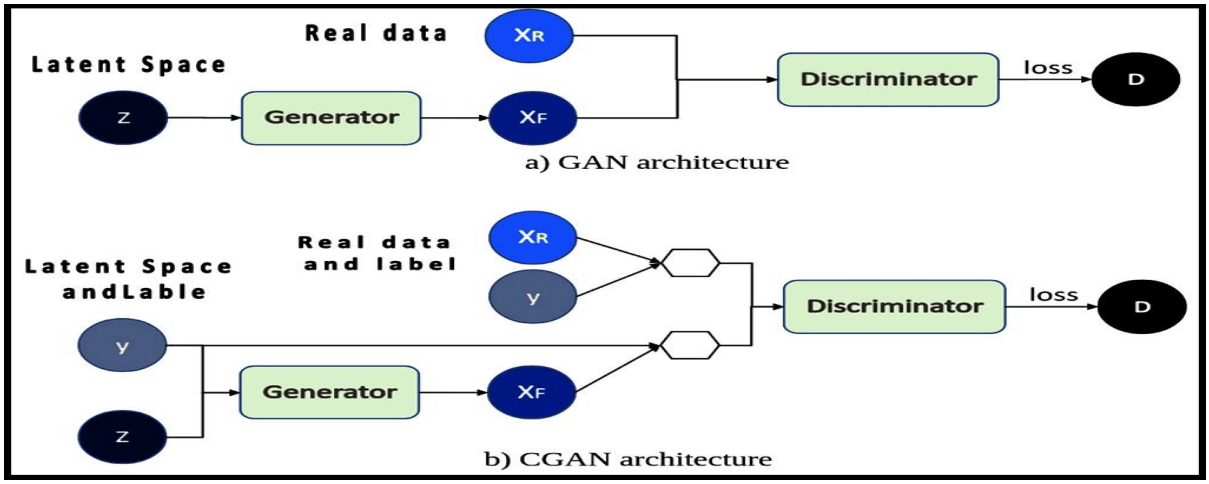


Figure 1- 17:Architecture standard de CGAN.**[43]**

C) WassersteinGAN (WGAN)

Le réseau génératif d'adversaire de Wasserstein **[44]** est désigné par l'acronyme WGAN, il a été introduit par l'auteur Martin Arjovsky dans leur article 2017 intitulé « Wasserstein GAN », son principe est de chercher une autre méthode pour l'amélioration et la stabilisation du modèle en vue de mieux approximer la distribution des données observées.

La différence entre le GAN et WGAN est que lors de la classification ou bien la prédiction d'une image générée, le principe de GAN se base sur le modèle de discriminateur avec une probabilité que cette image soient réelles ou fausses. Par contre le WGAN remplace le modèle de discriminateur par un critique qui détermine le réel ou le faux d'une image donnée.

L'objectif de WGAN est que la formation de générateur cherche à optimiser et à minimiser la distance entre la distribution des données observées dans l'ensemble de données d'apprentissage et la distribution observée dans les exemples générés. Tout s'est prouvé à l'aide d'un argu-

ment mathématique, qu'on le considère constant selon différentes mesures de distance de distribution, tel que : la divergence de Kullback-Leibler (KL), Jensen-Shannon (JS), la distance Earth-Mover (EM), appelée distance de Wasserstein.

La seule différence entre ces distances et leur impact se résident dans la convergence des séquences de distribution de probabilité. WGAN est plus stable et sensible à l'architecture demodèle. Le plus important à noter, est que la perte du discriminateur semble être liée à la qualité des images créées par le générateur. **« Voir la figure 1.18 »**

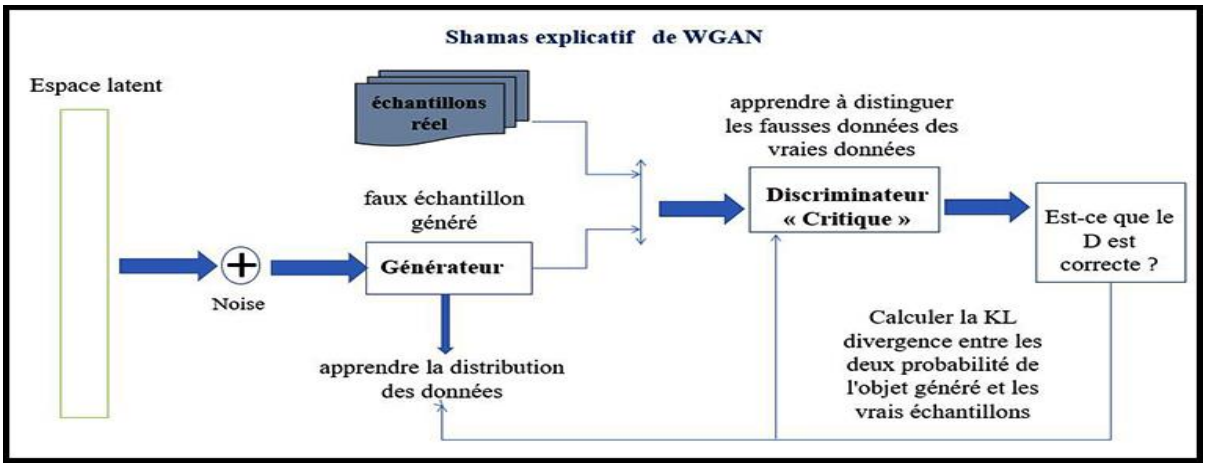


Figure 1- 18:schèma représente l'architecture de WGAN .

❖ **Divergence de Kullback-Leibler (KL divergence)**

La divergence de Kullback-Leibler **[45]** son acronyme (divergence K-L), appelé aussi entropie relative, elle est inventée en 1950 par les deux cryptanalystes américains Solomon Kullback et Richard Leibler**.**

Cette divergence représente une mesure de dissimilarité entre deux distributions de probabilités **P** et **Q**, tel que :

- Le **P** représente une distribution de probabilités calculée avec précision.
- Le **Q** représente une description ou une approximation de **P** (distribution de réponses attendues).

La **KL** est calculée par cette fonction :

Dans le cas où les **P** et **Q** deux distributions de probabilités discrètes, on utilise

La 1ere fonction : la somme de produit scalaire entre p(i) et **P** par rapport à **Q**.

$$DKL^{(p\|Q)} = \sum_{i} p(i) \log \frac{p(i)}{Q(i)} \qquad (9)\textbf{[45]}$$

Dans le cas où les **P** et **Q** sont des variables continues, on utilise une intégrale (2éme fonction)

$$D_{kL}(P_{\|}Q) = \int_{-\infty}^{\infty} p(x) \log \frac{P(x)}{Q(x)} dx \quad (10)$$ **[45]**

- **D (P || Q)** représente combien **Q** est relativement différent pour **P**.
- **D (P || Q)** = 0 Aucune différence entre la description **P** et **Q** .
- **D (P || Q)** proche de 0 il est très similaire à la description réelle.
- **D (P || Q)** égale à 1 il est peu similaire à la description réelle.

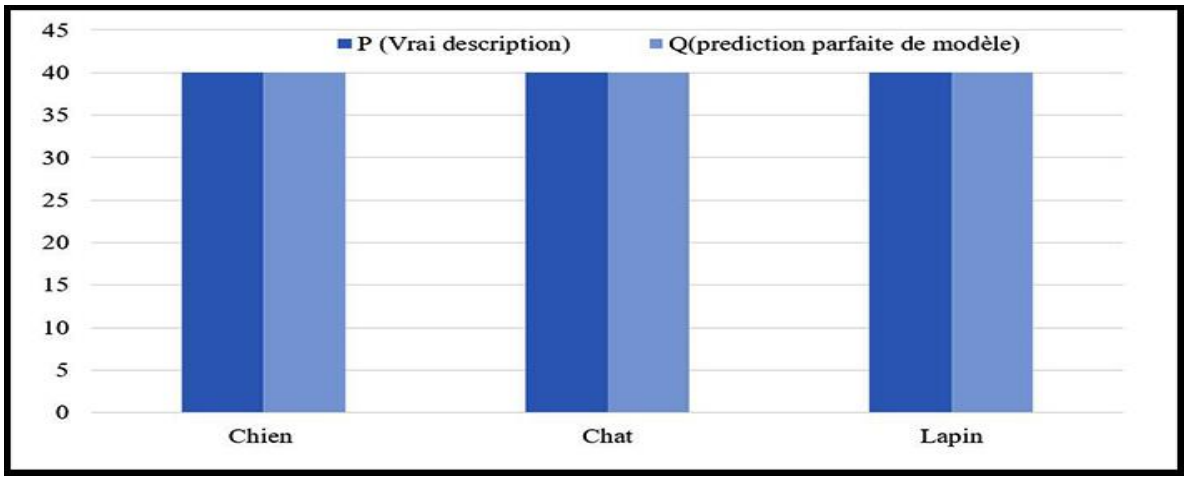


Figure 1- 19: représente la similarité entre les deux descriptions de probabilité.

- Dans la **(figure 1-19)**, on remarque une similarité totale entre la vraie description **P** et la prédiction de model **Q**. on trouve ces résultats dans le cas où **D (P || Q)** = 0. Dans les trois catégories (chien, chat, labin).
- Dans la **(figure 1-20)**, on remarque une petite différence de similarité entre la vraie description **P** et la prédiction de model **Q**. on trouve ça dans le cas où **D (P || Q)** proche de 0 (la catégorie chien et chat). Par rapport à la catégorie des lapins, il y a une grande différence entre **P** et **Q** on trouve cette situation dans le cas où **D (P || Q)** égale ou proche de 1.

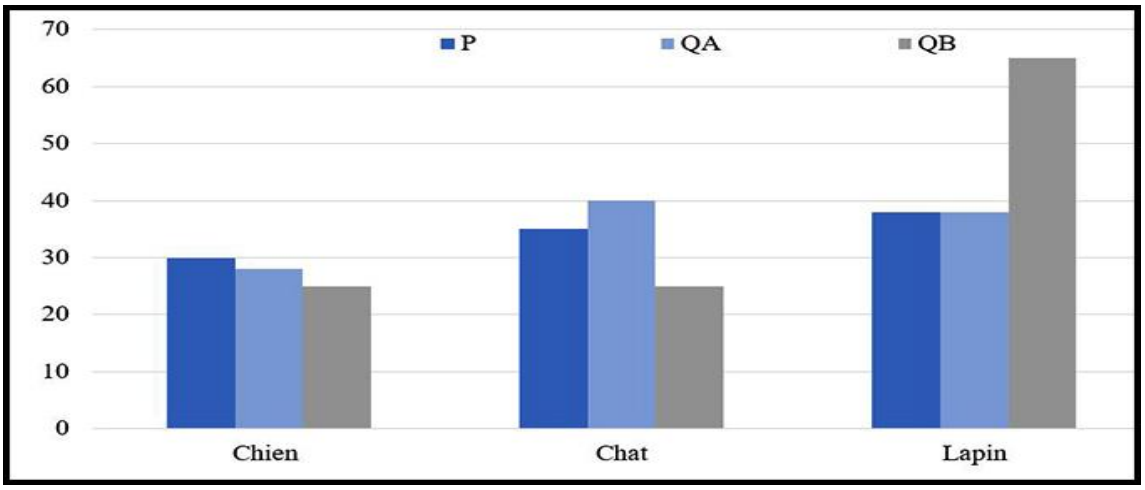


Figure 1- 20:graphe représente la différence de similarité entre P,QA,QB.

2.4. Test de performance d'un modèle en Apprentissage Automatique

Les tests de performance **[46]** consistent à évaluer les performances d'un système en termes de réactivité et de stabilité sous une charge de travail particulier. Les tests de performance sont généralement exécutés pour examiner la vitesse, la robustesse, la fiabilité et la taille de l'application, et aussi dans quelle mesure un système peut-il séparer les documents pertinents des documents non pertinents pour une requête utilisateur donnée **[47].**

La bonne manière de mesurer la performance **[48]** de notre modèle de Machine Learning est de tester celui-ci sur des données qui n'ont pas servi à l'entraînement, c'est à dire il faut entraîner notre modèle sur une partie seulement du Dataset (les données de notre base de données) et utiliser la seconde partie pour évaluer la vraie performance de notre modèle. On appelle cela le *Train set* et le *Test set*.

- **Train set et Test set :**

On divise ainsi le Dataset aléatoirement en deux parties avec un rapport 80/20 :

- **Train set (80%) :** qui permet à la machine d'entraîner un modèle.
- **Test set (20%) :** qui permet d'évaluer la performance du modèle.

Pour créer un «Train set » et « Test set » à partir de notre Dataset, on utilise la fonction train_test_split de Sklearn, On peut ensuite entraîner notre modèle sur le $\boldsymbol{Train\ Set}$: ($\boldsymbol{xtrain}$, $\boldsymbol{ytrain}$), puis l'évaluer sur le $\boldsymbol{Test\ Set}$:($\boldsymbol{xtest}$, $\boldsymbol{ytest}$).

Donc, pour évaluer le test set il existe 4 méthodes : **Accuracy, précision**, **Recall** et **F1 score**.

Chaque métrique mesure quelque chose de différent sur les performances du système, pour cette raison, il est souvent souhaitable d'optimiser et de prioriser une métrique par rapport à une autre.

La métrique d'optimisation dépend du contexte et des objectifs du système. Pour qu'on puisse calculer ces quatre mesures il faut d'abord construire une matrice qui s'appelle la matrice de confusion.

❖ **Matrice de confusion**

La matrice de confusion **[49] (voir le tableau 1-1)** ou tableau de contingence s'agit d'un outil utilisé pour mesurer les performances des modèles de **« Machine Learning »,** tout en examinant spécifiquement la précision des prédictions par rapport au résultat réels dans les problèmes de classification sur un ensemble de données de test.

	Résultat réel		
Résultat prédit		Positive	Négative
	Positive	**Vrai positive**	**Faux positive**
	Négative	**Faux négative**	**Vrai négative**

Tableau 1- 1:représente la matrice de confusion.

Tout ce qui est présenté en vert **(voir le tableau 1-1)** résume les classifications **correctes** effectuées par le système .

Tout ce qui est présenté en rouge résume les classifications **incorrectes** effectuées par le système .

Tel que :

1. **Vrai positif** : ce sont les cas où la prédiction est positive, et la valeur réelle est effectivement positive (résultat correcte d'une prédiction positive).
2. **Faux positif** : ce sont les cas où la prédiction est positive, mais la valeur réelle est négative. (une prédiction positive incorrecte).
3. **Vrai négatif** : ce sont les cas où la prédiction est négative, et la valeur réelle est effectivement négative (résultat correcte d'une prédiction négative).
4. **Faux négatif** : ce sont les cas où la prédiction est négative, mais la valeur réelle est positive (une prédiction négative incorrecte).

Les performances correctes se produisent lorsque le système produit des vrais positifs ou des vrais négatifs. Des performances incorrectes se produisent lorsque le système produit des faux positifs ou des faux négatifs.

❖ **Accuracy**

L'Accuracy **[50]** répond à la question suivante :

Dans l'ensemble, à quelle fréquence notre modèle est-il correct (ou Combien d'interprétation le système a-t-il correctement classé ?

L'accuracy est simplement un rapport des classifications correctement prédites (à la fois les vrais positifs + les vrais négatifs) à l'ensemble de données de test total **(voir équation(12)).**

$$Accuracy = \frac{vrais\ positives + vrais\ negatives}{total\ exemples} \quad (12)\textbf{[50]}$$

L'accuracy est une excellente mesure, il peut nous dire immédiatement si un modèle est correctement entraîné et comment il peut fonctionner en général. Cependant, il ne donne pas d'informations détaillées sur son application au problème, alors c'est pourquoi on ne peut pas compter seulement sur elle.

❖ **Précision**

La précision **[50]** répond à la question suivante :

Lorsque le modèle prédit positif, à quelle fréquence est-il correct ?

La précision est le rapport des résultats générés par le système qui ont correctement prédit des observations positives (vrais positifs) au total des observations positives prévues du système, à la fois correctes (vrais positifs) et incorrects (faux positif) **(voir équation (13))**.

$$Précision = \frac{vrais\ positives}{vrais\ positives\ + faux\ \ positives} \quad (13)\ \textbf{[50]}$$

Un modèle qui ne produit aucun faux positif a une précision de 1,0. En d'autres termes, la précision est la capacité de notre modèle à ne pas déclencher d'alarme que pour un vrai incendie, donc cela aide lorsque les coûts des faux positifs (fausses alarmes) sont élevés.

❖ **Recall (Rappel)**

Le Rappel **[50]** répond à la question suivante :

Parmi toutes les réels positives, combien de celles-ci le système a-t-il correctement classé comme vrais positives ?

Le rappel est le rapport des résultats générés par le système qui ont correctement prédit les observations positives (vrais positifs) à toutes les observations réels positifs (vraie positives et faux négatives) **(voir équation (14)).**

$$Recall = \frac{vrais\ \ positives}{vrais\ positives\ + faux\ \ negatives} \quad (14)\ \textbf{[50]}$$

❖ **Score F1 (F-mesure)**

Le score F1**[50]** est la moyenne pondérée de la Précision et du Rappel (Recall). Par conséquent, ce score prend en compte à la fois les **faux positifs** et les **faux négatifs** pour trouver un équilibre entre précision et rappel **(voir équation (15))**.

$$F1Score = 2 * \frac{Prècision * Recall}{Prècision\ + Recall} \quad (15)\ \textbf{[50]}$$

Un excellent score F1 signifie que votre taux de faux positifs et de faux négatifs sont tous les deux faibles, ce qui vous permet d'identifier correctement les menaces réelles sans être dérangé par de fausses alarmes. Un score F1 de 1 est considéré comme parfait et un modèle de 0 est un échec complet.

Remarque : tous les modèles sont faux, mais certains sont utiles. En d'autres termes, tous les modèles généreront des faux négatifs, des faux positifs et peut-être les deux en même temps. Bien que vous puissiez ajuster le modèle pour minimiser l'un ou l'autre, vous rencontrerez souvent un compromis en ce qu'une diminution des faux négatifs conduit à une augmentation des faux positifs, et vice-versa. Donc, vous devrez optimiser les mesures de performances les plus utiles pour votre problème spécifique.

Conclusion

En conclusion, nous disons d'une manière générale que notre premier chapitre a été articulé en premier lieu sur l'intelligence artificielle, celle-ci a été basée sur deux parties nécessaires :

l'apprentissage automatique avec ses trois types et les réseaux de neurones artificiels génératifs adversaires. D'après cela, nous avons pu de saisir et de comprendre que :

Le CNN est dédié pour le traitement (des images, formes...) et le RNN est dédié pour le traitement de texte en concentrant sur ses deux modèles LSTM et GRU qui sont dédiés pour la conservation des informations précédentes.

À la fin, notre travail a été articulé sur la performance d'un system de machine Learning, et sur la spécification de bonnes manières pour avoir testé ses performances.

Pour le chapitre suivant on va entamer à la présentation de l'état de l'art, c'est-à-dire les travaux connexes qui sont en relation avec notre approche.

II. Chapitre

État de l'art « les travaux liés à la génération de la forme 3D à partir du texte.»

1. Catégorie texte en image.
2. Catégorie image en forme.
3. Catégorie texte en forme.

II. Chapitre : les travaux liés à la génération de la forme 3D à partir du texte»

Introduction

Ce chapitre sera consacré pour l'explication de notre état de l'art , tel que, on va citer tous les travaux connexes qui sont en relation avec notre problématique et en plus on va classifier ces derniers en trois catégories principales :

- les travaux de la catégorie "Texte en Image" .
- les travaux de la catégorie "Image en forme" .
- les travaux de la catégorie "Texte en forme".

1. Etat de l'art

À partir de cette partie, nous visons à mettre en lumière les travaux théoriques représentant le socle de notre recherche, qui consiste à générer une forme 3D à partir d'une description textuelle. D'après les recherches scientifiques présentées déjà dans notre premier chapitre, nous avons pu saisir que la chose qui nous aide pour générer par exemple : une image, un texte ou bien une forme ...c'est les réseaux génératifs adversaire "GAN".

Pour atteindre notre objectif, nous avons basé sur nombreux travaux antérieurs, dont les premiers, sont ceux qui concernent la catégorie (Texte en Image) :

1.1. Texte en Image

Généralement, les recherches scientifiques liées à cette catégorie ont pour but la génération des images à partir de texte, ils prennent en considération les caractéristiques des images qui doivent être générées.

Au cours des dernières années, des progrès importants ont été réalisés dans l'apprentissage des intégrations visuo-sémantiques, parmi ces progrès nous citons, l'apprentissage nul (zéro-shot Learning) qui a été caractérisé par la génération automatiquement des légendes d'images pour des images Web générales.

Dans leur article « *Learning deep representations of fine-grained visual descriptions* », Reed et al,**[51]** ont essayé de résoudre le problème de la compréhension de l'image et de mettre en relation des images et le texte, en devenant les pionniers de cette approche (texte en images).

D'après ces auteurs, le problème de la mise en relation des images et du texte est encore loin d'être résolu en particulier pour fine-grained régime, donc afin de résoudre ce problème ils ont utilisé une préformation sûr des grands ensembles de données d'images (ils ont utilisé deux

ensembles de données de descriptions visuelles à fine-grained : un pour l'ensemble de données d'oiseaux **Caltech-UCSD**, et un autre pour l'ensemble de données de fleurs **Oxford-102**).

Au cours de cette préformation, les auteurs ont basé sur des travaux antérieurs portants essentiellement sur : « les modèles de langage au niveau des caractères » et « l'apprentissage zéro à granularité fine (fine-grained zéro-shot Learning) », et ce en vue de former des encodeurs de texte de grande capacité à partir de zéro pour intégrer conjointement des descriptions visuelles et des images à granularité fine appelée (Deep Structured Joint Embedding).

Afin de représenter les descriptions visuelles à grain fin, Reed et al **[51]** ont utilisé un nouvel réseaux hybride convolutionnel récurrents(Convolutional Récurrent Net (CNN-RNN)), ce modèle a eu comme avantage la capacité de régler le problème de traitement du texte en image.

En résume de tout cela, nous disons que les auteurs Reed et al,**[51]** ont eu comme résultat que le réseau de neurones hydriques CNN-RNN se dote de deux fonctionnalités, les CNN sont dédiés pour la compréhension des éléments fins de l'image et les RNN sont dédiés pour la compréhension de la sémantique de texte.

Après quelques mois de la même année de leur premier article, Reed et al ont amélioré la méthode, cette fois leur recherche est orientée vers la génération automatique de l'image à partir de texte. De ce fait, dans leur article, intitulé *« Générative Adversarial Texte To image Synthesis »*.**[52]** les auteurs ont présenté une méthode pour la synthèse automatique d'images réalistes à partir d'une description textuelle, ils ont affirmé que leur objectif de générer des images réalistes à partir de texte est intéressant et utile au même temps.

Au début, ils ont trouvé des difficultés pour réaliser cette approche à cause de l'ancienneté de la technique de IA, mais après quelques années et avec l'évolution des architectures des réseaux de neurones récurrent comme les RNN et les réseaux génératifs GAN, elle est devenue très fonctionnelle, capable pour apprendre des caractéristiques de texte et générer des images plausibles avec haute qualité.

Après, ils ont testé leur modèle avec les deux jeux de données (Dataset): l'ensemble des données des oiseaux **« Caltech-UCSD Birds »** et l'ensemble de données fleurs **« Oxford-102 Flowers»** réalistes, en traduisant ces derniers par des phrases sous forme de description textuelle.

Dans ce contexte, les auteurs ont déduit que les réseaux de neurones convolutionnels récurrents ont une capacité de produire des représentations textuelles très discriminantes et au même temps généralisables, pour apprendre d'une manière automatique des mots et des caractères.

L'objectif de ce travail consiste à transformer directement les mots et les caractères aux pixels dans l'image, mais les auteurs ont assuré qu'il faut régler tout d'abord deux problèmes, à savoir :

- **Le premier** est relatif avec la nécessité de faire une représentation d'une fonction pour que le texte devient capable de capturer les détails visuels importants.
- **Le deuxième** est relatif avec l'utilisation de ces fonctionnalités pour réaliser une image plus réaliste et convaincante.

La question qui se pose ici, est comment ces auteurs ont pu arriver pour résoudre ces deux problèmes?

Dans ce sens, ils ont déclaré que l'apprentissage profond était une solution pour réaliser leur travail, cependant, ils ont affirmé qu'il reste un autre problème à soudre, est que, comment les images conditionnées apprendre seules une description textuelle multimodale ?

Dans le cadre de proposer une solution à ce problème, l'amélioration de l'apprentissage profond a permis de faire une décomposition séquentielle des mots ou des caractères, ce qui rend le modèle capable de prédire le prochain jeton conditionné sur l'image et tous les jetons précédents pour une prédiction bien définie.

Selon Goodfellow et al **[53]** a multimodalité conditionnelle est une méthode utilisée pour les réseaux contradictoires (GAN), tel que l'objectif de générateurs se concentre sur la génération des images synthétiques réelles pour tromper le discriminateur. C'est pour cette raison, Reed et al **[52]** ont conditionné à la fois le générateur et le discriminateur sur les informations secondaires que doivent être générées.

En résume de tout cela, nous disons que les auteurs Reed et al **[52]** ont réussi pour la génération automatique des images réaliste grâce au développement des architectures simple conditionné " CGAN ". La question qui se pose ici : est-ce que l'image générée est-elle de bonne qualité ou non ?

Après une année de travail de Reed et al **[52]** une autre méthode a été suggérée pour résoudre ce problème, cette dernière a été publié dans l'article *« Stack GAN: Texte To Photorealistic Image Synthesis with Stacked Generative Adversarial network »* de Han Zhang et al **[54]**ont déclaré que la composition d'une image à partir d'une description textuelle avec une meilleure qualité dépend un grand nombre d'applications.

Pour eux, les GAN sont des réseaux capables de générer des images liées au sens du texte, mais il est difficile de les former pour générer des images à haute résolution. Pour régler ce

problème, ils ont ajouté plus de couche de suréchantillonnage (multicouche) dans des modèles GAN.

À chaque fois l'image possède plus de détails, cette dernière trouve des difficultés au niveau de sa génération avec une grande résolution. Dans ce sens, Reed et al**[52]** ont réussi à générer des images 64 × 64 conditionnées par des descriptions textuelles, plausibles, mais ces dernières restent manquantes de certains détails (objet vif), pour cela ils ont ajouté des annotations supplémentaires dans les images de taille 128 × 128.

Han Zhang et al, ont affirmé aussi, qu'il existe des échantillons générés par l'approche texte en images, peuvent refléter le contexte général donné d'une description, mais sans déterminer les détails nécessaires d'un objet vif. ils ont ainsi proposé Stacked Réseaux Adversaires Génératifs (Stack GAN) qui représente une technique permet de générer des images 256 × 256 conditionnées par des descriptions textuelles.

Pour obtenir leur objectif clé, ils ont utilisé le processus d'amélioration de la qualité d'image par la décomposition en "stage-GAN", tel que :

- **Le Stage-I GAN :** détermine la forme et la couleur primitives. dans cette première partie, la résolution des images générées est très basse.
- **Le Stage-II GAN :** utilise les résultats de stage 1 comme une entrée. Cette partie représente la partie corrective, du fait qu'elle est consacrée pour rectifier les défauts de résultat de stage 1, générer des images réalistes avec haute résolution et pour ajouter plus de détails convaincants.

La nouvelle technique de conditionnement par les "Stack GAN" proposée par Han Zhang et al , était une solution qui a permis d'augmenter la diversité des images synthétisées, et selon la comparaison avec les travaux précédents, ils ont démontré que cette méthode atteint des améliorations sur la génération des photo-réalistes conditionnées aux descriptions textuelles.

Après une année, il y avait un autre travail qui a été proposé dans le même contexte qui prend en considération les détails fins d'une image et qui focalise sur les mots pertinents dans le texte. Ce travail a été présenté dans l'article de Tao Xu et al **[55]** , intitulé *« AttnGAN: Fine-Grained Text to Image Génération with Attentional Générative Adversarial Networks ».*

Tao Xu et al a proposé un nouveau Framework par rapport aux autres travaux de même domaine de la génération texte en images, tel que leur méthode permet de créer un réseau alternatif attentionnel génératif (Attn Gan), avec ce dernier ils ont pu de synthétiser les détails fins

d'une image avec une grande attention sur les éléments pertinents « mots » dans la description textuelle.

Pour entrainer le générateur, ils ont proposé un modèle de similarité multimodale attentionnelle profonde qui permet de calculer la perte de correspondance entre le texte et l'image, ils ont ainsi testé leur résultat sur deux jeux de données et ils ont déclaré que cette méthode a signalé environ de 14,14% sur l'ensemble de données **« CUB »** et 170,25% sur l'ensemble de données **« COCO »** les plus difficiles.

En effet, l'analyse de l'auteur « Tao Xu et al » montre que les couches l'AttnGAN sont capables de détecter automatiquement les conditions appliquées au mot, afin de générer les différentes parties d'images.

Les auteurs de la majorité des travaux qui se basent sur la synthèse texte en image ont utilisé les réseaux génératifs d'Adversaires (GAN), tel qu'ils ont codé les descriptions textuelles en des vecteurs, en considérant ça comme une condition pour générer l'image. Mais ces auteurs ont affirmé que s'ils utilisent le vecteur de phrase global, le résultat généré est une image de moins qualité, et ce à cause de la non-détermination dans le vecteur quel sont les mots pertinents (les informations importantes) qui doivent être générés, surtout dans les scènes complexes, c'est pour cela ils ont utilisé l'architecture globale d'Attn Gan.

Leur modèle comporte deux composants :

- Le premier composant est un réseau attentionnel génératif, où le mécanisme d'attention est développé pour rendre le générateur dessine les parties importantes de l'image selon les mots pertinents indiqués et le vecteur global de phrase, tel que chaque mot soit encodé par un vecteur, l'objectif de cette première partie est de générer une image de basse résolution.

Pour interroger les vecteurs des mots dans l'étape suivante, il suffit d'utiliser le vecteur d'image dans chaque sous-région, et ce pour l'entrainer sur le contexte des mots grâce à l'utilisation de couche d'attention. À la fin, les deux vecteurs d'image régionale et contexte de mots sont combinés pour la création du contexte multimodal vecteur, cette partie garantir une image haute résolution avec plus de détails.

- Le deuxième composant est le Deep Attentional Modèle de similitude multimodale (DAMSM), son objectif est permet de calculer la similarité entre l'image générée et la phrase, et de fournir la perte de correspondance entre images en texte pour la formation de générateur.

Leur méthode est triple :

1) Elle utilise les composants AttGAN et DAMSM pour synthétiser l'image à partir une description textuelle .

2) Elle surpasse les modèles GAN au niveau de sa technique.

3) Elle est capable de détecter les mots pertinents pour la génération des images, et ce grâce à l'utilisation des GAN conditionnel en couches.

Selon l'auteur Tao Xu et al **[55]**, nous avons constaté que dans les dernières années, le mécanisme d'attention a été utilisé avec succès dans la traduction automatique aux plusieurs niveaux pour l'explication des images. À leur connaissance les GAN sont utilisés seulement dans l'étape après traitement sans attention, et les l'AttnGAN sont orientés à aider les GAN pour générer les éléments fins des images à haute qualité au niveau des mots et phrase.

Nous concluons, qu'à travers les travaux présentés dans la première catégorie, les auteurs ont atteint leur objectif avec succès dans la génération des images à partir de texte. Maintenant on passe à la deuxième catégorie :

1.2. Image en forme

Dans cette partie on va voir comment constituer une forme à partir d'une image.

Dans l'article « *3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction* »,Choy et al **[56]**ont reconstruit des formes 3D à partir des images 2D, pour cela, ils ont proposé une nouvelle architecture de réseau neuronal récurrent qui s'appelle 3D réseau neuronal de reconstruction récurrente **(3D-R2N2)** (est une extension du LSTM standard).

Le réseau apprend un mapping d'une ou plusieurs images d'objets à leurs formes 3D sous-jacentes à partir d'une grande collection de données synthétiques (Shape Net, PASCAL 3D, online Products, MVS_CAD_Models), mais la différence entre ce travail et les travaux précédents, est que leur réseau ne nécessite aucune annotation d'image ou étiquette de classe d'objets pour la formation ou les tests.

Ce travail permet donc la reconstruction d'objets 3D dans des situations où les méthodes (SFM / SLAM) traditionnelles échouent (en raison du manque de texture et / ou d'une large ligne de base) (mauvaise qualité) .

L'objectif du réseau est de réaliser des reconstructions 3D mono vue et multi-vue. L'idée principale est de profiter de la puissance du LSTM pour conserver les observations précédentes et affiner progressivement la reconstruction de la sortie au fur et à mesure que de nouvelles observations deviennent disponibles.

La fonction d'erreur du réseau utilisée est définie comme la somme de cross-entropie au niveau du voxel.

On comprend que Choi et al **[56]** ont réussi à reconstruire des formes en 3D à partir des images 2Ds par l'utilisation de l'approche 3D-R2N2 .

Après quelques mois de la même année, Jiajun Wu et al **[57]** ont publié un article intitulé «*Apprentissage d'un espace latent probabiliste de formes d'objets via la modélisation 3D GAN »,* où ils ont étudié le problème de la génération d'objets 3D.

Les chercheurs ont fait des progrès impressionnants dans la modélisation et la synthèse d'objets 3 D, principalement basés sur **meshes** ou **skeletons**. La majorité de leurs méthodes traditionnelles, synthétisent des nouveaux objets en empruntant des pièces à des objets dans les bibliothèques de modèles CAD existantes. Par conséquent, les objets synthétisés semblent réalistes, mais pas conceptuellement nouveaux.

C'est pour cette raison, ils ont proposé un nouveau cadre à savoir, 3D Generative Adversarial Network (3D-GAN), qui génère des objets 3D à la fois nouveaux et réalistes, et ce, à partir d'un espace probabiliste en tirant parti des avancées récentes dans les réseaux convolutifs volumétriques et les réseaux adversatifs génératifs. Et aussi, ils ont introduit une nouvelle 3 D-VAE-GAN comme une extension de 3D-GAN, cette extension génère les objets 3D à partir des images 2 D, ceci est inspiré par VAE-GAN proposé par l'auteur larsen et al , qui consistent à combiner VAE et GAN en partageant le décodeur de VAE avec le générateur de GAN.

Donc, nous disons que l'auteur Jiajun Wu et al **[57]** ont réussi à générer une nouvelle forme 3D sans l'emprunt des pièces d'objets de la bibliothèque CAD, et ce par l'utilisation d'extension de 3D-VAE-GAN de 3D-GAN.

D'après tous les travaux qu'on a déjà vu sur la génération Texte en image et image en forme, on présente dans ce qui suit, la troisième catégorie qui est la base de notre recherche.

1.3. Texte en forme

Dans l'article *« Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings »* Chen et al **[58]** ont présenté une méthode pour la génération des formes en 3 dimensions à l'aide d'une description textuelle à partir de langage naturel.

Pour ce faire, ils ont débuté par deux études importantes :

la première concerne la compréhension de la forme libre de la description textuelle .

la deuxième concerne la compréhension des formes en 3D colorées.

L'intérêt de leur modèle a été focalisé sur l'approche d'apprentissage métrique, qui serve à combiner ces deux dernières études, et ce afin de produire une représentation conjointe entre le langage et les caractéristiques physiques (la couleur, la longueur, largeur…) des formes 3D.

Pour évaluer leur approche, ils ont basé sur la collecte d'un ensemble de formes 3D avec leur description textuelle en langage naturel (Shape Net).

La transformation et la génération de ces formes 3D à partir des descriptions textuelles ont requis l'intégration d'un nouveau framework appelé CWGAN (conditionnel wasserstein génératif adversal network). Par rapport à eux, cette transformation(texte en forme) représente la première façon utilisée pour connecter les textes à l'objet 3D.

L'objectif d'utilisation de langage naturel, est que:

La langue est une façon de communication des idées, et le domaine de IA (l'intelligence artificielle) est basée sur la création d'un lien entre le langage naturel et les modalités visuelles.

Les auteurs Chen et al **[58]**ont exprimé tout ça par une petite expérience, qui consiste à présenter d'un ensemble d'objets de « shapeNet » sur quelques individus, et les demande de décrire la forme 3D sous une description textuelle, par l'exemple *: «table basse ronde en verre avec quatre pieds en bois»*. l'objectif de ce travail est de générer une forme à partir de ce petit texte d'une manière concrète visuelle.

Dans ce contexte, les auteurs ont parlé de deux tâches principales sont : la récupération et la génération du texte en forme. L'objectif de la récupération texte en forme est de permettre d'utiliser des bases de données contenant un ensemble des formes sans revient à l'explication d'humaine pour représenter une forme donnée.

le deuxième objectif est un objectif clé, car il représente vraiment leur problématique de travail, tel que la conception en 3D est un domaine qui dépend beaucoup plus des matériaux et des logiciels très puissants, comme (Maya, balander…), mais le problème est que la majorité de ces derniers sont couteux et en plus qu'il y a des gens qui ne maitrisent pas la conception en 3D du fait qu'elle est manuelle, fastidieuse et très longue, dépend une grande durée de travail.

Donc, leur travail a été une solution majeure pour les concepteurs 3D tels qu'ils ont profité de cette technique pour gagner le temps.

Ils ont parlé de trois parties principales :

A) La description textuelle et la forme 3 D: cette partie permet d'exploiter un nouvel ensemble de descriptions textuelles relatives à un ensemble des formes 3D coloré, choisis dans l'étude, tel que (les tables des chaises...).

B) La méthode d'apprentissage par association et métrique: dans la deuxième partie ils ont expliqué comment apprendre conjointement un texte, et regroupé les formes 3D similaires a une description textuelle à partir d'une connexion sémantique implicite (en pointillées lignes).

C) Dans la troisième partie qui est aussi divisée sur deux tâches principales :

- **C1 :** *« la récupération de texte en forme »* : cette phase, consiste à ramener toutes les formes 3D relatives qui peuvent appartenir à cette description textuelle, car, parfois la même forme peut avoir plusieurs descriptions textuelles, ou parfois le contraire à cause que chacun peut présenter la forme ou bien le texte à sa manière.
- **C2 :** « la génération de texte en forme »: cette partie peut être nommée la partie générative du fait qu'elle génère une nouvelle forme 3D à partir d'une nouvelle description textuelle.

En vue d'atteindre leur objectif de relier et de connecter le langage naturel à une forme, ils ont eu besoin d'utiliser un système qui est capable de comprendre le langage naturel et en même temps les formes 3D. de ce fait, ils ont utilisé un espace pour faire la jointure entre les deux.

À leur connaissance, ils ont affirmé *« qu'il n y a pas un travail avant qui a parlé sur l'intégration de texte en forme 3D »*.

Les auteurs des approches précédentes ont focalisé sur l'utilisation des étiquettes et les attributs, mais le problème est que, cette approche ne détermine pas sur quelle base ils ont classé les objets, est-ce que selon la couleur, matériaux ou bien le style?

L'objectif de Chen et al **[58]** ne s'intéresse pas à l'utilisation des attribue ou bien des étiquettes, mais il s'intéresse à améliorer la façon qui permet de faire l'intégration conjointe de texte et de forme d'une manière directe, ça veut dire à partir d'une description textuelle qui génère directement la forme 3D.

Cet objectif est très important, mais ils ont trouvé des difficultés au niveau de sa réalisation, car il n'y a pas un lien unique entre la forme et le texte, de plus, ils ont trouvé que plusieurs-sieurs descriptions textuelles correspondent à la même forme, et même chose pour le cas contraire .

En effet, dans ce dernier article, plusieurs méthodes ont été présenté dans le but résoudre ces problèmes, parmi elles :

- une méthode d'apprentissage d'un texte et de forme, qui peut transformer d'une manière directe la description textuelle vers une forme 3D. La structure de cette méthode est basée sur la formation de texte et de forme, de bout en bout, où les auteurs ont essayé d'associer les points

similaires, et de donner à la fois une modalité, que ce soit texte vers texte ou bien forme vers forme, et même entre textes en forme, ce qui nous intéresse le plus.

- Une méthode d'apprentissage par association entre les descriptions qui permet d'établir les liens entre les descriptions et les formes.
- Une méthode d'apprentissage métrique qui consiste à renforcer les liens entre texte et forme.

Pour la récupération et la génération de la forme à partir de texte. Les auteurs ont utilisé le réseau conditionnel Wasserstein générative adversaire WCGAN, qui est un réseau hybride entre le WGAN et CGAN :

- le WGAN : assure la stabilité de réseau et règle le problème de GAN.
- le CGAN : permet d'attribuer une condition au réseau.
- Le WCGAN : ajoute un niveau de qualité et une diversité, et en plus il donne le bon rondement par rapport à l'utilisation l'un des deux .

2. Résultats obtenus

Articles	Catégorie	Approche	Point fort	Point faible
[51]	Texte en image	CNN-RNN	Dédiés pour la compréhension des éléments fins de l'image et la sémantique de texte.	Il n'est pas utilisé pour générer des images à partir de texte
[52]	Texte en Image	CGAN	Génère des images réalistes à partir de texte avec des caractères souhaitables grâce à la condition ajouté au niveau du générateur et discriminateur.	N'assure pas la qualité de génération.
[54]	Texte en Image	Stack-GAN	Améliore la qualité des images générées	Ne capture pas les détails fins de l'image
[55]	Texte en Image	AttGAN	Synthétiser les détails fins de l'image avec une grande attention sur les éléments pertinents (mots) de la description.	Synthétiser les détails fins de l'image à haute qualité avec une grande attention sur les éléments pertinents (mots) de la description.

[56]	Image en forme	3D-R2N2	Apprends un mapping d'une ou plusieurs image d'objet à leurs forme 3D à partir de dataset	Les images converties en forme 3D ne sont pas nouvelles, sont des images de (dataset).
[57]	Image en forme	3D-VAE-GAN	génère des nouvelles forme sans l'importation des objets de Bibliothèque CAD.	Ne garantit pas la qualité des objets générer
[58]	Texte en forme	CWGAN	Générer des formes 3D à partir de la transformation des mots pertinents de texte en voxèl colorée.	Parfois générer des formes de mauvaise qualité

Tableau 2- 1:représente les résultats obtenus par les trois catégories.

Conclusion

On conclura notre deuxième chapitre par un petit rappel sur tous ce qu'on a déjà vu dans cette partie. Notre ensemble des travaux connexes sont scindés en trois catégories principales : la catégorie texte en images, la catégorie image en forme, et la catégorie Texte en forme.

On a compris que la première classe visait à générer des images à partir de descriptions textuelles, elle a choisi environ quatre méthodes (CNN-RNN, CGAN, Stack Gan et Att Gan), à chaque fois on note qu'il y a des points ajoutés au niveau de l'approche par rapport a la précédente.

La deuxième catégorie a pour but de transformer des images en formes, elle prend en considération ces deux approches (3D-R2R2, 3D-VAE-GAN).

D'après la classification des travaux qu'on a faite on a compris que les deux premières catégories ont donné naissance de la troisième catégorie texte en forme qui est la base de notre travail tel que cette dernière elle utilise l'architecture de CWGAN . Dans ce qui suit, on va passer à la partie conceptuelle, qui sera consacrée pour exprimer l’architectures et les méthodes utilisées dans la réalisation de notre approche .

III. Chapitre

La conception détaillée de la génération texte en forme

(Architecture et les méthodes utilisées)

1. Architecture générale.
2. Traitement de texte et la forme.
3. Récupération de texte en forme.
4. Génération de texte en forme.

III. Chapitre 3 :la conception « la partie pratique »

Introduction

Dans ce chapitre, on va passer à la partie la plus importante dans notre travail, c'est la partie conceptuelle. Cette dernière elle sera consacrée pour la présentation de notre méthode de génération de formes 3D colorées à partir du langage naturel.

Notre chapitre est divisé en deux parties importantes :

Premièrement on va parler sur la récupération de texte en forme, plus précisément sur l'incorporation conjointe (joint embedding) de la description de texte en forme libre et des formes 3D colorées. De ce fait, la présente partie porte essentiellement sur la présentation de notre modèle qui se base sur la combinaison entre l'apprentissage par association et les méthodes d'apprentissage métrique.

deuxièmement on va parler sur la génération de textes en forme, plus précisément on va déterminer l'architecture générale et l'architecture de CWGAN qu'on a utilisé dans notre approche

La tâche de génération de texte en forme 3D colorée et un nouveau travail, pour cette raison, nous allons concentrer d'une manière importante sur tache, car la plupart des descriptions de formes impliquent des propriétés de couleur ou de matériaux.

1. L'architecture général

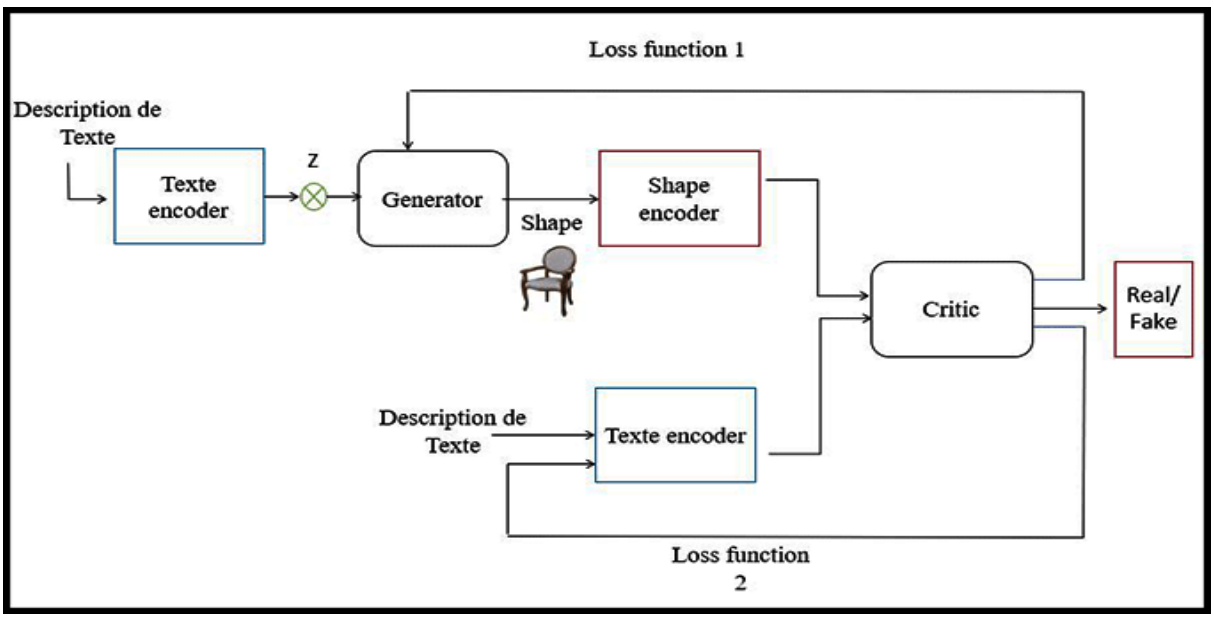


Figure 3- 1:L'architecture général de notre approche .

La **« figure 3-1 »** montre l'architecture générale de la conception de notre modèle en entrée de générateur, les descriptions textuelles convertir sous forme d'une représentation vectorielle via un encodeur de texte en utilisant la méthode (d'encodage a chaud et word2 avec)

Ensuite, le générateur génère la forme 3D correspondant en fonction du vecteur. Le générateur est basé sur une méthode d'apprentissage en profondeur et un cadre GAN.

Lorsque le texte vectorisé est entré dans le générateur, il sera concaténé directement avec le vecteur de bruit z « noise vecteur z » pour assurer la flexibilité du générateur.

La sortie sera une forme 3D sous la forme d'une grille de voxel, qui est ensuite convertie en représentation vectorielle par un encodeur de forme (Shape encoder).

La sortie du codeur de forme est directement connectée à l'entrée du réseau critique avec une description textuelle.

Le critique se connecte à deux fonctions de perte, une pour la formation des générateurs (si le critique a bien classé la forme) et l'autre pour la formation des discriminateurs (si le critique a mal classé la forme). Alors le réseau critique tente de calculer avec précision la distance Wasserstein (la fonction perte / l'erreur). En revanche, le générateur tente de minimiser la distance calculée par le critique en rétro propagé l'erreur calculée

2. Traitement de texte et de forme

Afin de pouvoir entraîner notre modèle sûr de telles données, nous devons d'abord et avant tout définir un format d'entrée pour arrive à la sortie désirée, nous allons donc d'abord encoder nos textes et formes en 3D.

2.1.Traitement de texte

Dans cette partie, nous allons montrer les processus nécessaires au traitement de texte. Premièrement, nous convertissons nos descriptions de texte en vecteurs en utilisant la méthode d'encodage rapide, puis nous transmettons les vecteurs au modèle CNN-RNN, et en sortie nous obtenons les vecteurs pour la fusion de texte (inclure du texte).

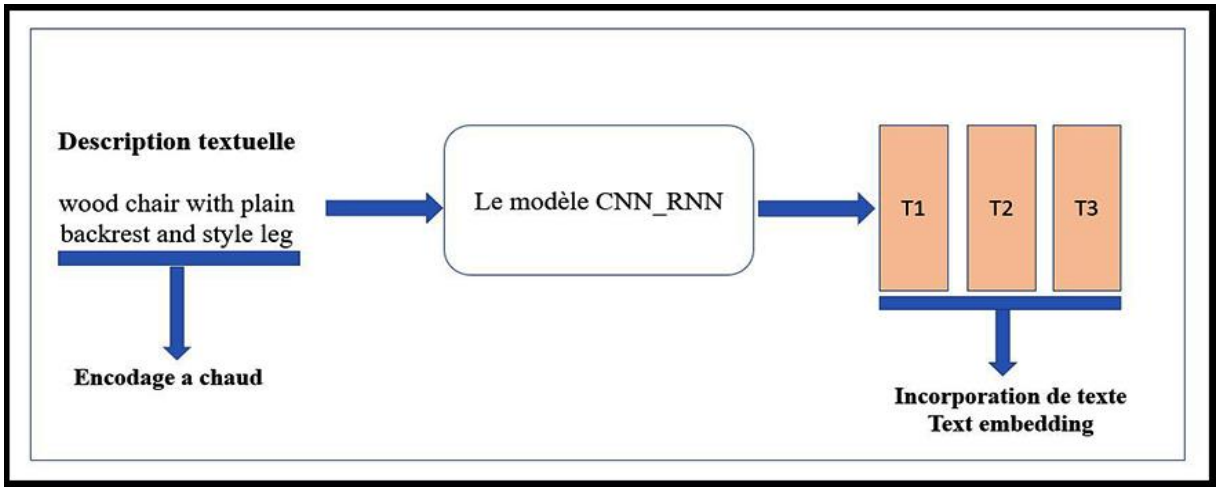


Figure 3- 2:schéma générale de l'encodage de texte.(notre travail d'après). **[58]**

2.1.1. Encodage de texte (Text encoder)

Avant d'utiliser les descriptions textuelles dans n'importe quel modèle, la première des choses qu'on a faite est de vectoriser ces descriptions. Cette phase est appelée incorporation de texte.

Les réseaux de neurones prennent des chiffres en entrée dans leur traitement et non pas un texte. Dans la littérature, deux méthodes sont proposées pour convertir du texte en données numériques :

La première est une méthode traditionnelle : l'encodage à chaud.

La deuxième est une méthode basée sur l'utilisation des vecteurs de mots (word2vec).

Dans les parties qui suivent, on va expliquer chacune d'elles en détail.

A) L' encodage à chaud

Pour transformer notre description textuelle en représentation vectorielle on utilise la méthode d'encodage a chaud **[59]** , le principe de la méthode et permet de transformer chaque mot de la description textuelle en vecteur binaire V (valeurs appartiennent à l'ensemble {0,1}), tel que ;

Le V : est le nombre total des mots dans la phrase.

- La valeur 1 : représente l'index de la position du mot par rapport la description textuelle.
- La valeur 0 : représente l'index du reste des mots.

par exemple: la description « Deep Learning is hard fun »:

si nous voulons coder le mot Deep qui se trouve dans la première position de la phrase, nous marquons la valeur 1 dans la position 1, et nous marquons la valeur 0 dans le reste des positions

(0 pour les positions des mots : Learning, is, hard, fun) et ainsi de suite. **« voir la figure 3-3»**

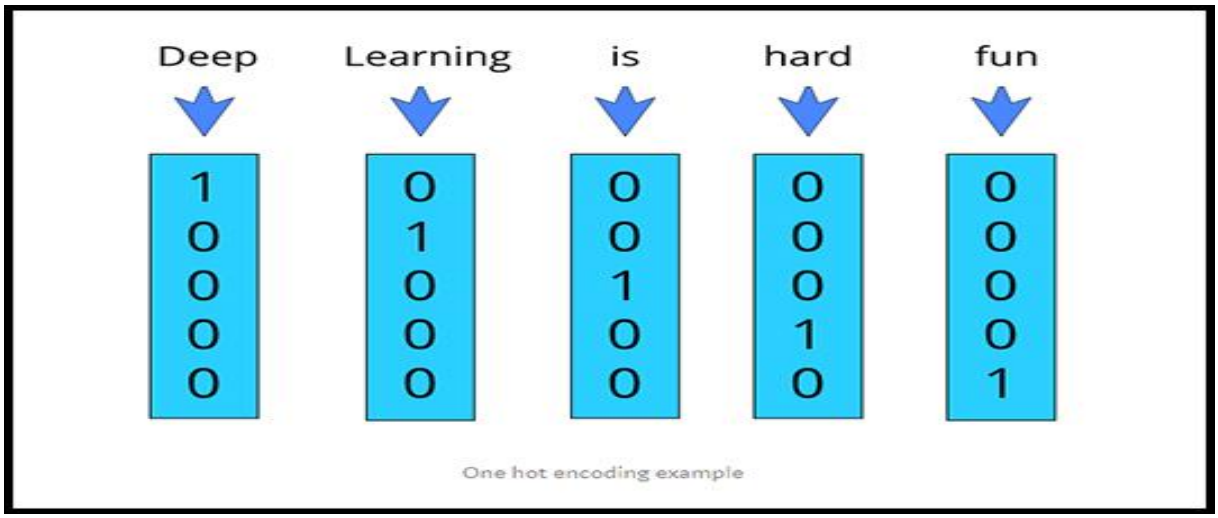


Figure 3- 3:schéma représente la transformation de texte en vecteur.**[59]**

B) Vecteurs deux mots (word2Vec)

Pour indiquer quels mots se produisent à proximité d'un certain mot on prend on considération la transformation vectorielle de la description textuelle par encodage a chaud puis on applique la méthode Word2Vec nous choisissons le modèle skip Gram de Mikolov **[60]** nous utilisons quelque chose appelée la fenêtre contextuelle.

B-a) Le modèle Skip Gram

Pour prédire tous les mots de contexte à partir d'un mot cible, dans notre cas nous utilisons le modèle Skip Gram**[60].**

Dans cette partie nous avons utilisé un réseau de neurones pour la prédiction, tel que:
L'entrée sera une version codée à chaud du mot cible.
La sortie sera une seule version codée a chaud pour le mot contexte. **V**: c'est la taille des couches d'entrée et de sortie (nombre de vocabulaire).Entre les deux couches précédentes il y a une seule couche cachée, est celle qui détermine la taille des vecteurs de mots qu'il faut avoir à la fin. Ce principe est expliqué dans le modèle suivant, **« voir la figure 3-4»**

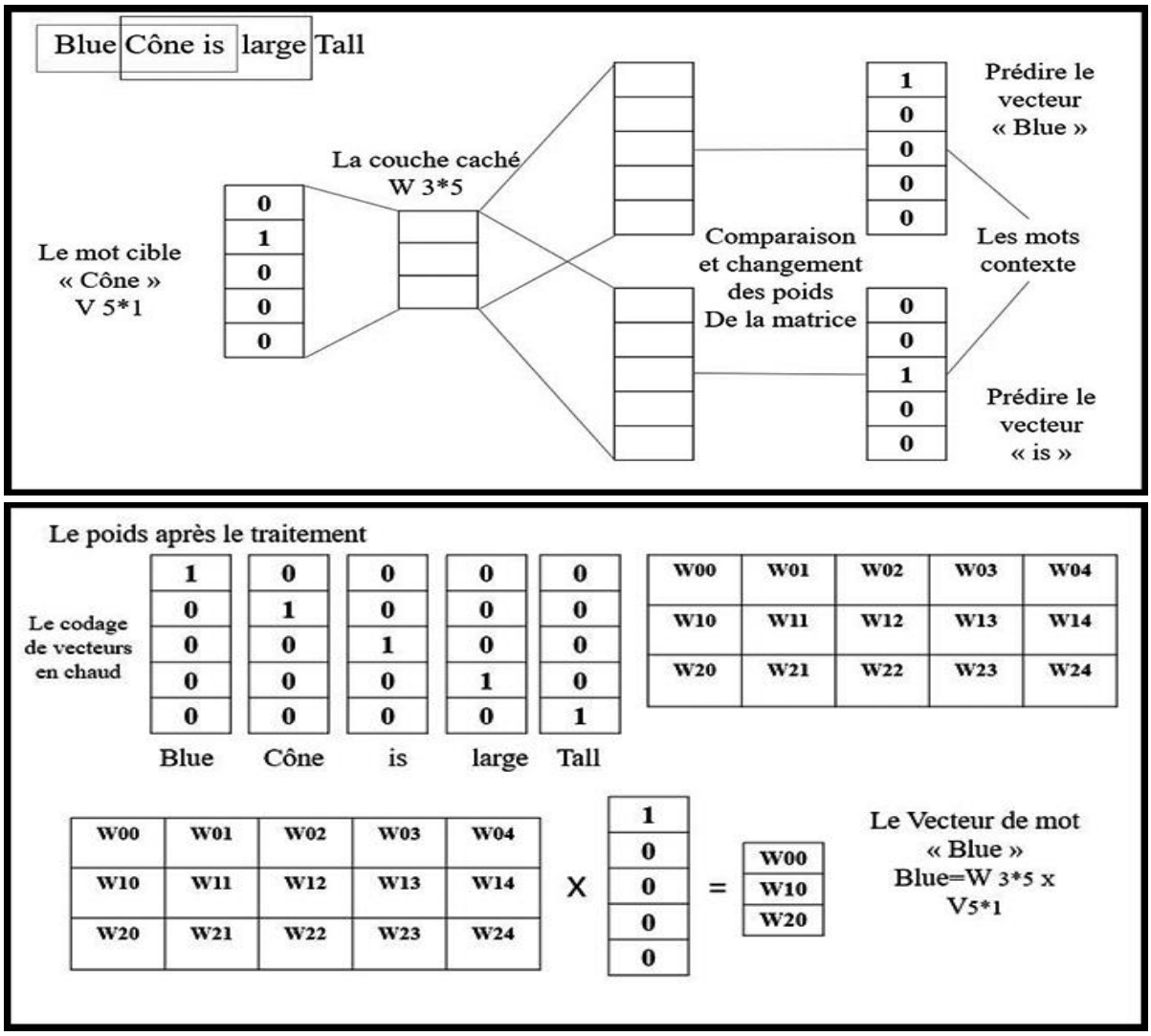


Figure 3- 4:schéma générale de modelé Skip Gram.

2.1.2. Incorporation de texte (Text embedding)

Dans cette partie, nous montrons comment nous utilisons le CNN pour l'extraction d'entités textuelles et comment nous utilisons le RNN(GRU) pour la prédiction et la classification multi-étiquettes . Avant d'utiliser le modèle CNN-RNN, tout d'abord nous préformons le modèle word2Vec , en vue de capturer les fonctionnalités de chaque mot puis la formation de CNN-RNN(GRU),tel que nous utilisons le CNN pour l'extraction d'entité pertinente de texte saisi en entrée, et à la fin un vecteur de fonctionnalité sera présenté comme un "état initial" pour la connaissance préalable du RNN .

et nous utilisons le RNN pour la prédiction de la séquence d'étiquettes.

A) Le model CNN_RNN

Dans cette partie on va expliquer comment transformer les vecteurs des mots en incorporation (embedding) par l'utilisation de modèle hybride CNN_RNN (GRU), tel que :

- Le CNN servira à faire l'extraction des caractéristiques de texte.
- Le RNN (GRU) servira à faire la prédiction de séquence des mots.

❖ **Architecture de CNN_RNN**

L'architecture de notre modèle est présentée comme suit :

à l'entrée du modèle, un vecteur de taille (3588*128) représente la description de texte qu'il a été vectorisé par l'utilisation des méthodes (encodage à chaud, word2avec.) la concaténation de vecteurs de mots doit donner le texte original. Les données analysées sont transmises aux cartes d'entités, puis ils seront stockés dans un emplacement spécifique dans la couche convolution - elle qui suit :

On passe premièrement par 4 couches de convolution (la partie qui représente le CNN), tel que :

La première couche et de taille (128*128*3), la deuxième couche est de taille (128*128*3) suivie par une couche de batch normalisation, la troisième a 256 filtres de taille (128*3) et la quatrième a aussi 256 filtres de taille (256*3) suivie par une couche de batch normalisation. Après chaque convolution une fonction d'activation Relue est appliqué pour produire une carte de caractéristiques, cette dernière est dotée d'un avantage par rapport au sigmoïde et Tanh.
Ensuite, une fois la partie CNN est terminée sa sortie sera une entrée pour la deuxième partie RNN(GRU) avec 256 unités.

La sortie de GRU sera une entrée d'une couche fully connected (fc5) qui possède 256 noue-rons, puis une fonction d'activation Relue est activée. Ensuite, nous passons le résultat obtenu aux dernières coucher fully connected (fc6) qui possèdent 256 nouerons et qui délivrons fina-lement un vecteur à faible dimension de caractéristiques de texte, tous les poids ont une régu-larisation L2 avec un poids 0,0005.

2.2.Traitement de la forme (Shape embedding)

❖ **Qu'est-ce qu'une incorporation de forme (Shape embedding)**

Shape embedding **[61]** peut être considéré comme un paramétrage des formes en fonction de leurs caractéristiques les plus marquantes. Par exemple : bien que les formes des chaises ou

des tables différentes les unes des autres, nous aimerions trouver un moyen dès les présenter en fonction d'un petit nombre de leurs propriétés le plus marquant.

Plus formellement, l'intégration est une cartographie d'un espace de haute dimension vers un espace de faible dimension. Donc nous aimerions « encoder » toutes les formes en un petit vecteur seulement en utilisent 3D CNN **« voire la figure 3-5»**

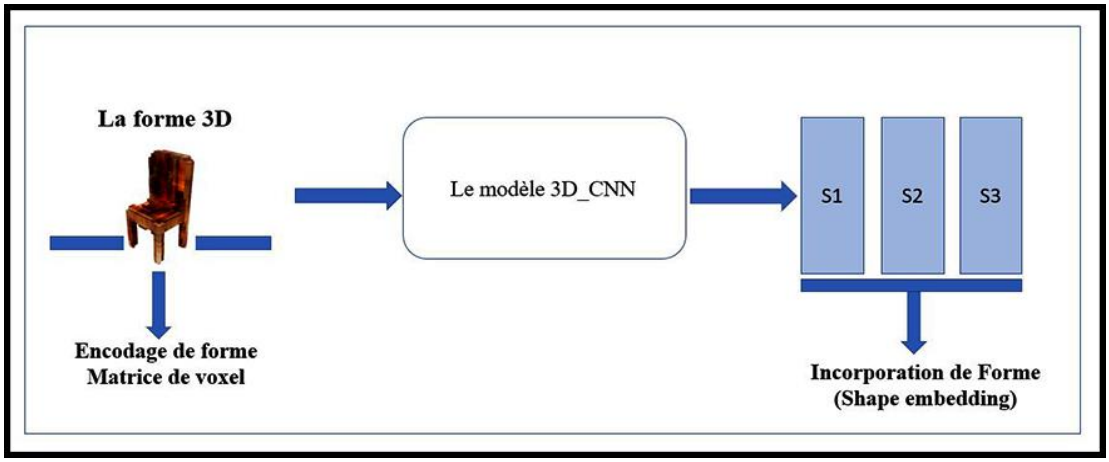


Figure 3- 5:schéma représente le traitement de la forme.(notre travail d'âpres). **[58]**

A) Architecture 3D-CNN

Notre architecture 3D CNN proposée utilise une matrice de voxel de taille (32*32*32*4) comme une entrée et la sortie à prévoir est une Shape embedding S.

L'architecture 3D CNN contient 3 blocs répétés d'une couche conventionnelle 3D (3 × 3 × 3) tel que :

Le premier possède 64 filtres de taille (16*16*16), le deuxième a 128 filtres de taille (8*8*8) et le dernier a 256 filtres (4*4*4) est suivi par une couche batch de normalisation, cette dernière permet de normaliser les valeurs de la couche d'entrée en ajustant et en mettant à l'échelle les activations.

La fonction d'activation utilisée et le Relu, celle-ci force les neurones à retourner des valeurs positives, ensuite nous passons à une couche d'avg-pooling (Average pooling) qui permet de renvoyer la moyenne des éléments sur une fenêtre de calcul. À la fin, nous terminons notre architecture par une couche de fully connected (fc5) de 128 nouerons et une softmax qui Permets de calculer la distribution des probabilités des classes sur les classes de sortie, tel que toutes les couches sont suivies d'un Relu sauf la dernière.

4. La récupération de texte en forme

4.1.Récupération de texte en forme

Pour atteindre l'objectif de récupération de texte en forme, nous avons besoin d'un système capable de comprendre le langage naturel et les formes 3D, et une façon de connecter le langage aux formes qui consistent à utiliser un espace d'incorporation conjoint pour le texte et la forme (joint embedding Space).

4.2.l'apprentissage de la représentation mixte de texte-en-forme 3D

Les formes réelles sont incorporations avec de nombreuses couleurs, matériaux et attributs géométriques, les travaux antérieurs ont été basé sur les catégories à grain fin pour apprendre une bonne intégration (embedding).

Dans cette approche, nous évitons d'utiliser les catégories d'attributs ou d'annotations à granularité fine, car elles sont très coûteuses. Par conséquent, nous utilisons des descriptions de texte au niveau de l'instance pour chaque forme (5 descriptions pour chaque forme) (Associations au niveau de l'instance), et aussi nous choisissons l'apprentissage métrique pour regrouper les formes avec les descriptions similaires, et à la fin nous séparons les descriptions et les formes qui sont différentes entre eux.

Cependant, les formes et les descriptions sémantiquement similaires doivent être associées en utilisant l'apprentissage par association, nous pouvons établir ainsi des liens latents entre des formes similaires et du texte.

Nous combinons ensuite l'apprentissage métrique avec l'apprentissage par association dans la perte multimodale. **« voir la figure 3-6 »**

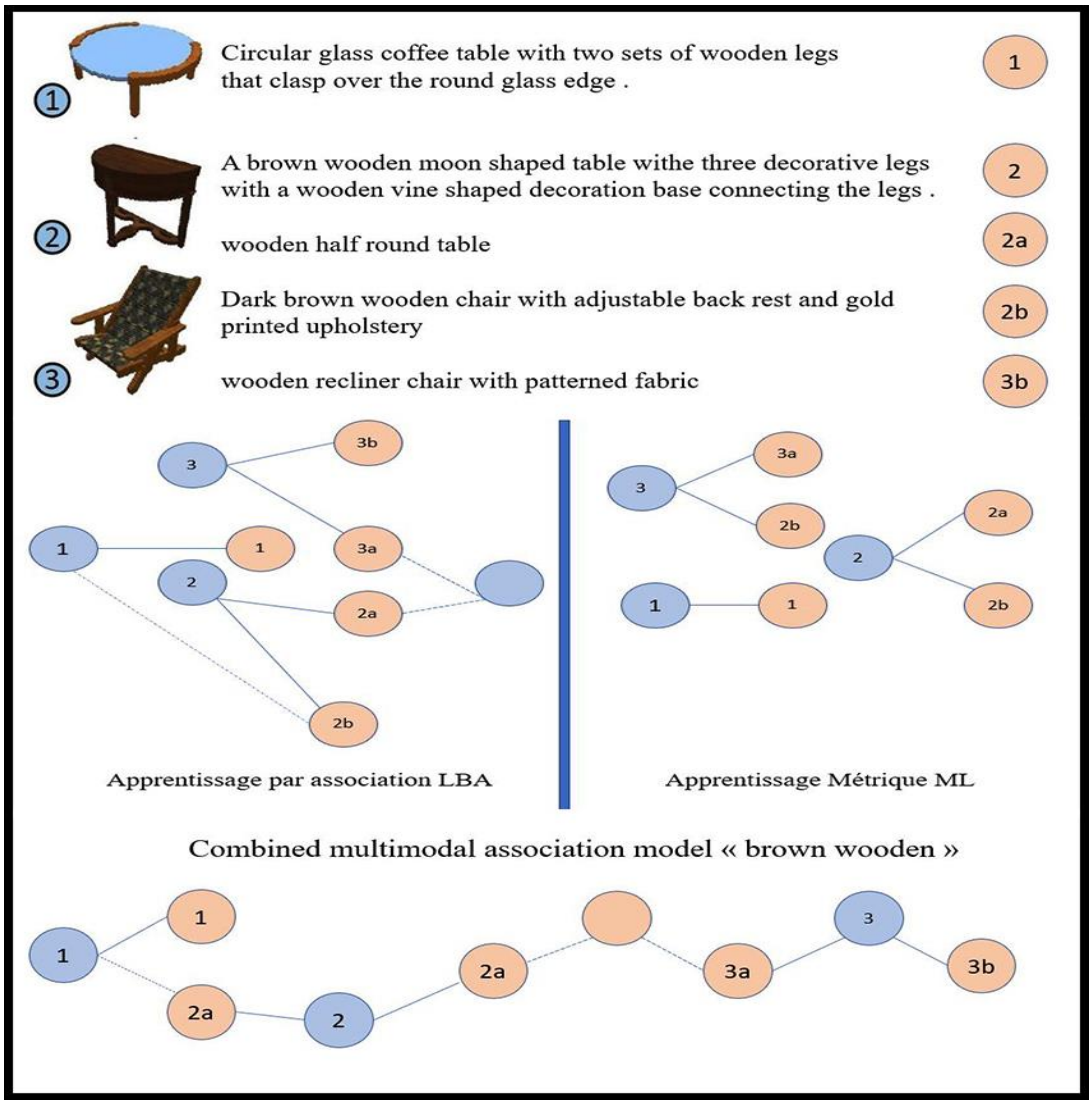


Figure 3- 6:représente la jointure de deux méthode ML et LBA.**[58]**

4.3.Regroupement de texte et de forme

Après l'encodage de texte et de forme, on va regrouper les textes similaires ensemble et les formes similaires ensemble. Pour ce faire, nous utilisons l'approche d'apprentissage par association qu'est inspirées de travaux récents de l'auteur P. Haeusser **[62].**

4.3.1. Le regroupement de texte en texte

Le principe de la méthode de récupération de texte en texte **[58]** consiste à récupérer les phrases voisines les plus proches entre elles, et ce, à partir le choix d'une phrase requête d'après ensemble de jeu de données Shape Net , dans ce sens, nous avons constaté que presque toutes les descriptions récupérées proviennent des formes différentes, mais elles sont sémantiquement similaires.

Par exemple: le modèle apprend que «**Ovale** » est similaire que « **Circulaire** ».

- **La requête :** « une table ovale marron avec base en trois sections » .
- **Les Voisins les plus proches :** « la table est une table de canapé avec un plateau ovale. Les pieds sont arrondis et la table est en bois multicolore. »
- « Table circulaire de couleur beige avec 3 supports où les deux premiers supports sont circulaires et la partie en Botton est rectangulaire et elle est supportée par une large pièce en bois en C. »

4.3.2. Le regroupement de forme en forme

Cette méthode se fonctionne sous le même principe que la récupération texte en texte seulement, cette fois nous remplaçons les textes par les formes, et nous récupérons les formes les plus proches en fonction de leur intégration apprise.

Les voisins sont souvent convenables à la requête dans le style et la catégorie, par exemple : lorsqu'une chaise est interrogée, les voisins proches récupérés sont également des chaises, toute en focalisant sur les attributs, (rectangulaire /ronde…) ou bien (table /chaise…).

4.4.Apprentissage des associations multimodales

4.4.1. Apprentissage par association (LBA)

Pour apprendre les associations de texte en forme intermodale : nous avons utilisé l'approche de Chen **[58],** nous supposons « n » forme avec « m » descriptions textuelle, la fusion de ces dernières ce fait à travers un codeur de texte nommé « T », et un autre codeur pour la forme nommé « S ».Premièrement nous définissons une matrice de similarité M_{ij} entre les vecteurs T et S (similarity between embeddings S and T) comme le produit comme le produit scalaire entre eux : dont des i^{ème} et j^{ème} ligne présenté dans la matrice.

$$Mij \equiv T_i . Sj \qquad (1)\ \textbf{[58]}$$

Après nous transformons cette matrice de similarité en probabilités en utilisant le soft max M .tel que cette dernière est calculée par équation 2 :

$$p^T i^S j = {exp(M_{ij})}/{\sum_{j'} exp(M_{ij'})} \qquad (2)\ \textbf{[58]}$$

Inversement, nous obtenons les probabilités de transition dans l'autre sens, P^{ST} en remplaçant M par M^T,et nous définissons la probabilité d'aller-retour (**round trip probability**) commençant de T_i et se terminant à T_j , elle est présentée par cette formule:

$$P^{TST}{}_{ij} = (p^{TS} p^{st})_{ij} = \sum_k p_{ik}^{TS}\ p_{ki}^{ST} \qquad (3)\ \textbf{[58]}$$

Nous utilisons la probabilité d'aller-retour pour associer la description i à certaines formes j.

Pour une description i donnée, notre but est de rendre ${P^{TST}}_{ij}$ uniforme sur la description j, qui est similaire à la description i. Par conséquent, nous définissons la perte aller-retour $\mathcal{L}_R^{TST}$ comme l'entropie croisée entre la distribution P^{TST} et la distribution uniforme cible.

Pour associer la description textuelle à toutes les formes correspondantes possibles, on impose également une perte sur la probabilité d'associer chaque forme à une description P^{Visit}

$${P^{Visite}}_i = \Sigma_i P_{ij}^{TS} / m \quad (4) \textbf{ [58]}$$

Pour maximiser l'entropie dans une description P^H,nous utilisons l'entropie croisé entre ${P^{Visite}}_j$ et la description uniforme sur les formes possible pour calculer la perte $\mathcal{L}_H^{TST}$ d'entropie. Donc la relation de perte d'association intermodale est calculé par équation (5):

$$\mathcal{L}^{TST} = \mathcal{L}_R^{TST} + \lambda\, \mathcal{L}_H^{TST} \quad (5) \textbf{ [58]}$$

4.5.Associations au niveau de l'instance

Pour l'apprentissage associatif, au début dans les travaux précèdent,ont utilisé les étiquettes au niveau des instances, car comme nous avons déjà vu précédemment, les étiquettes ne déterminent pas explicitement les formes et les descriptions non étiquetées pour les correspondances entre eux, donc ils ont utilisé l'apprentissage associatif cross modal.

Ce dernier est basé sur la supervision directe sur les relations trans-modules, il consiste à ajouter un avantage pour les associations trans-modules entre les descriptions et les formes qui n'étaient pas étiquetées pour apprendre mieux. Par exemple: (deux chaises similaires, dans ce cas il faut voir chacune à sa propre description textuelle).

Nous utilisons le tour de lancement additionnel. Cette façon améliore la performance dans le cas des problèmes multimodal, aussi elle respect la perte d'aller-retour de TST (texte-Shape-texte) $\boldsymbol{\mathcal{L}^{TST}}$, et STS (Shape-texte -Shape) avec la perte $\boldsymbol{\mathcal{L}^{STS}}$ qui peut prend une forme comme on a vue l'équations déjà.

$$\boldsymbol{\mathcal{L}^{STS} = \mathcal{L}_R^{STS} + \lambda \mathcal{L}_H^{STS}} \quad (6) \textbf{ [58]}$$

Par cette méthode, les auteurs garantissent que les formes possibles sont reliées à toutes les descriptions textuelles, en plus ils produisent des supervisions, ce qui aide l'ensemble des données qui ne possèdent que des étiquettes au niveau de l'instance.

4.6.Apprentissage métrique multimodal (ML)

Dans cette partie nous utilisons la deuxième méthode l'apprentissage métrique, dans cette dernière nous montrons comment déterminer la similarité entre les incorporations texte en texte et texte en forme avec l'utilisation de produits scalaires.

La méthode est présenté a l'aide d'un triplet de (x_i, x_j, x_K) qui intègre des descriptions textuelles, tel que les descriptions (x_i, x_j) noté (les paire positive) car les deux appartiennent à la même classe d'instance, et les descriptions (x_i, x_K) noté (les paire négative) car elles n'appartiennent pas à la même classe d'instance.

De ce fait, nous présentons la contrainte d'apprentissage métrique par l'équation **(7)**

$$F(x_i;\theta).F(x_j;\theta) > F(x_i;\theta).F(x_K;\theta) + \alpha \quad (7)\ \textbf{[58]}$$

Tel que le :

- F: représente l'espace .
- α : représente la marge.

Le produit scalaire des paires positives est supérieur au produit scalaires des paires négatives plus la marge α. la perte de texte en texte est présentée par l'équation (8).

$$\mathcal{L}_{ML}^{TT} = \frac{1}{2|P|}\sum_{(i,j)\epsilon\, P}\left[Log\,(v_i + v_j) - mi,j\right]_+^2 \quad (8)\ \textbf{[58]}$$

Tel que :

- mi,j: la similarité entrer x_i et x_j.

$$V_l = \Sigma_k \in N_l \exp\{\alpha + m_{l,k}\} \quad (9)\ \textbf{[58]}$$

Le N_i représente l'ensemble des indices négatifs qui appartiennent à une classe d'instance autre que la classe l, et P est un ensemble des indices positif tel que **i** et **j** appartiennent à la même classe d'instance.

Cette approche elle nous a permis de regrouper les descriptions textuelles qui sont similaires dans un ensemble et de séparer les descriptions textuelles différentes.

Pour appliquer la méthode de similarité entre texte en forme, nous appliquons le même principe que la similarité de texte en texte, sauf cette fois ci, nous utilisons :
les x comme des incorporations de textes et y comme des incorporations des formes, en plus nous n'oublions pas de calculer la perte de texte en forme de l'apprentissage métrique comme on a parlé déjà dans l'équation précédente. Equation (8) .

$$F(x_i;\theta).F(y_j;\theta) > F(x_i;\theta).F(y_K;\theta) + \alpha \quad (10)$$ **[58]**

4.6.1. Le principe de fonctionnent l'apprentissage métrique (ML)

Dans cette partie on va discuter sur l'apprentissage métrique qui est basé sur la mesure de similitude entre les objets ou bien (échantillons), pour faire cette dernière, il faut calculer la métrique de distance entre eux. Dans leur article, Mahmut KAYA **[63]** ont déclaré que l'apprentissage métrique approfondi offre une meilleure solution pour les données non linéaires grâce à l'utilisation aux fonctions d'activation.

La majorité des études utilisent les réseaux siamois et triplés pour établir la corrélation de similitude entre les échantillons. L'objectif de l'apprentissage métrique est basé sur la réduction de la distance entre les objets de même classe (similaire) et d'essayer de l'est rapproché, et augmenté la distance entre les objets de classe différente. Le rôle de la mesure de distance est de représenter les données qui ont plus de discrimination significative par l'utilisation de la relation de similitude entre les échantillons.**« voir la figure 3-7» .**

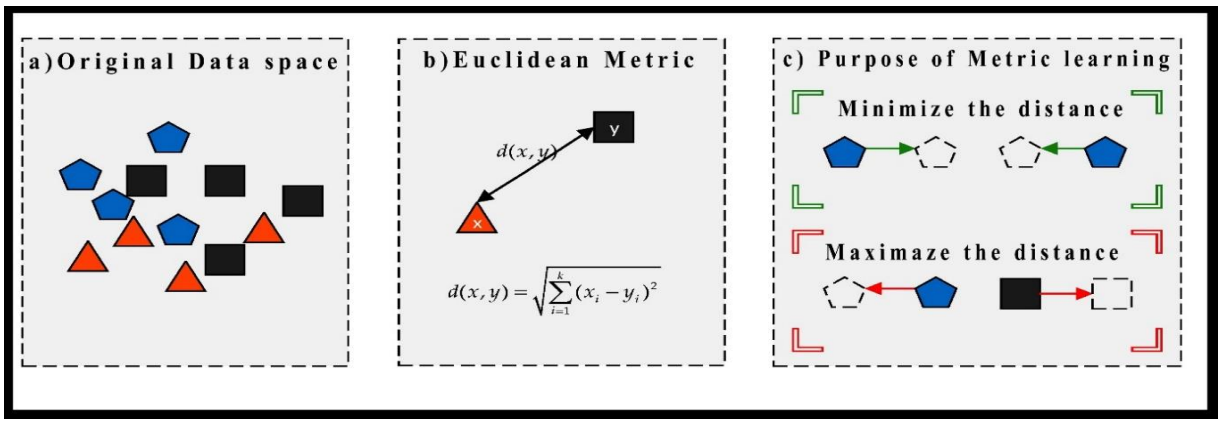


Figure 3- 7:schéma explicatif sur le l'apprentissage métrique **.[63]**

❖ **Triplet-similitude de perte (triplet similarity loss) :**

Dans les parties précédentes on a expliqué comment les descriptions textuelles transformées en vecteur de nombre (codé), par contre de cette partie sont consacrées pour savoir calculer la similarité entre les échantillons proposés dans notre cas (texte en texte, texte en forme ...).

Comme on a déjà parlé, après les textes convertis en nombre, on doit calculer la distance entre eux. La perte de triplet est considérée comme la fonction de perte pour les réseaux de neurones artificiels , elle est nommé triplet à cause de ces trois composants :

➢ **Composant 1:** une entrée de base nommé l'ancre (A), dans notre cas «le texte réel ».

- **Composant 2:** une entrée positive (P), dans notre cas « un texte qui est presque similaire au texte réel» .
- **Composant 3:** une entrée négative (N), dans notre cas « un texte qui est diffèrent au texte réel ».

Le triplet est noté (A, P, N), nous utilisons la distance euclidienne entre deux vecteurs notés :d(A,P) et d(A,N) pour restreindre le triplet. Cette dernière est calculée :

$$d(A,P) = \sqrt{\sum_{i=}^{N}(a_i - p_i)}^{2} \qquad d(A,N) = \sqrt{\sum_{i=}^{N}(a_i - n_i)}^{2} \tag{11}$$

Le principe de base est de minimiser la distance entre l'ancre (A) et l'entrée positive (P), et de maximiser la distance entre l'acre (A) et l'entrée négative (N), cette méthode est généralement utilisée dans le but d'apprendre les incorporations comme les incorporations des mots (Word embedding), elle est représentée par cette façon :

$$Triplet = \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha \tag{12}$$

Tel que :

- $d(A,p) = \|f(A) - f(P)\|^2$ et $d(A,N) = \|f(A) - f(N)\|^2$
- α: est la marge empirique ,la valeur a = 0,8.

La condition nécessaire de cette fonction est que : la différence entre les deux distances doit être inférieure strictement a 0.

- $Triplet = \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 < 0$

c'est à dire :

- $Triplet = \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha < 0$

Par exemple si les résultats de calcul de distance ont été obtenus comme ceci :

$d(A,p)$=0.5et $d(A,N)$=0.51 sans prendre en considération la valeur de la marge, nous serons considérés comme des résultats approximatifs (ne sont pas des bons résultats), par contre si on obtient $d(A,p)$=0.3 et $d(A,N)$=0.50 avec une marge α =0.2, les résultats serons considères comme bons.

4.7.Perte multimodale complète

Dans cette partie nous combinons les deux pertes d'apprentissage d'association et métriques pour déterminer la fonction finale de pertes **[58]** telle que :

$$\mathcal{L}_{Totale} = \mathcal{L}^{TST} + \mathcal{L}^{STS} + \gamma(\mathcal{L}_{ML}^{TT} + \mathcal{L}_{ML}^{TS}) \quad (13)\ \textbf{[58]}$$

5. La génération de texte en forme

5.1.La comparaison entre différents modèles de GAN

Pour la réalisation de notre approche de génération, nous choisissons quatre modèles :

- **Le premier (GAN)** : selon plusieurs auteurs **[57][64][65]**,ce modèle est très adapté à la génération de textes en forme, il encourage à générer une sortie avec des attributs corrects tout en évitant l'utilisation de contraintes per-voxel. L'importance de ce modèle se réside dans le fait que la même description doit pouvoir générer différentes formes qui capturent tous les attributs spécifiés.
- **Le deuxième (cGAN) :**selon l'auteur Mirza**[66],** ce modèle est très adapté à la génération de la donnée en fonction de sa condition, son objectif consiste à générer des formes qui capturent les attributs corrects, mais ces derniers ne correspondent pas à une instance de modèle spécifique qui serait incorrecte.
- **Le troisième (WGAN):**selon l'auteur Arjovsky **[67]**, ce modèle est très adapté à l'amélioration de la diversité de sortie tout en évitant les problèmes d'effondrement des modes (mode collapse) avec les GAN traditionnels.
- **Le quatrième (CWGAN) :** selon l'auteur Chen**[58],** le modèle représente la combinaison de trois sous modèles : (GAN, cgan et WGAN), il est considéré comme la base de notre travail du fait que notre objectif est de générer non seulement des formes qui capturent les attributs corrects, mais de générer la forme qui correspond à une instance (description) donnée, ceci est régler par le GAN conditionnel qui joue un rôle clé dans notre structure texte en forme (texte-To-Shape).

5.2.Les modèles choisis dans la génération de textes en forme

Parmi tous les modèles de GAN proposés pour la génération de textes en forme nous choisissons le troisième modèle **«CGAN»** et le quatrième **«CWGAN»** tel que le CGAN est une architecture déjà utilisé dans les travaux précédents et par rapport à notre projet on a utilisé l'architecture de CWGAN car est un modèle hybride qui regroupe tous les avantages des autres modèles précédents (GAN, CGAN, WGAN).

5.2.1. Architecture de génération de formes CGAN

Selon l'architecture de CGAN déjà mentionnée, CGAN est une sorte de GAN, mais la seule différence entre les deux est que dans CGAN, une condition est ajoutée au niveau du générateur et du discriminateur.

❖ **Le générateur :** En entrée de générateur il y a deux couches présentées, l'un pour le texte embedding ("x") de taille 136 et l'autre pour la condition ("y") qui possède les caractéristiques que nous voulons générer à la sortie. Les deux vecteurs sont d'abord passés à la première couche de convolution transposée, cette couche se compose de 256 filtres et d'une taille de 4*4, chacune de nos couches de convolution est suivie par une couche de batch normalisation et d'une fonction d'activation Relue. Cette fonction force les neurones à retourner des valeurs positives, ensuite les sorties de ces deux couches seront concaténées et 512 features map de taille 4*4 sera créée.

Les 512 features map qui sont obtenus auparavant, ils seront donnés en entrée de la deuxième couche de convolution transposée qui est composée aussi de 256 filtres de taille 8*8, une couche de batch normalisation et une fonction d'activation Relue sont appliquées sur cette couche, ensuite, nous répétons la même chose sur la couche de déconvolution trois, qui est composée de 128 filtres de taille 16*16, une batch normalisation et une fonction d'activation Relu sont appliquées toujours sur chaque déconvolution. la dernière couche de convolution transposée est composée d'un seul filtre de taille 32* 32 et la fonction d'activation Tanh est appliquée.

❖ **Le discriminateur :** comporte deux couches d'entrée, l'une est pour la sortie obtenue du générateur après le codage qui est de taille (1*32*32) et l'autre est pour la condition de taille (10*32*32), ces deux vecteurs passent d'abord à la première couche de convolution, elles sont composées de 64 filtres d'une taille de taille 16*16, suivis d'une couche de normalisation par lots (batch normalisation) et d'une fonction d'activation Leaky Relu,après avoir concaténé les résultats obtenus à partir de ces deux couches des 128 features map de taille 16*16 seront créés. Ensuite ces 128 features maps passés par 3 couches convolution (Conv2, Conv3, Conv4) tel que la taille de la première (Conv2) est (256*8*8) et la taille de la seconde (Conv3) sont (512*4*4), les deux sont suivies d'une couche de batch normalisation et d'une fonction d'activation « Leaky Relu », et après le Con4 qui est suivi par la fonction d'activation sigmoïde de sorte que la valeur de sortie soit limitée de 0 à 1, un vecteur de taille 1 sera obtenu, indiquant si la forme générée par le générateur est réelle ou fake. **« Voir la figure 3-8»** ***(voir l'annexe N°10).***

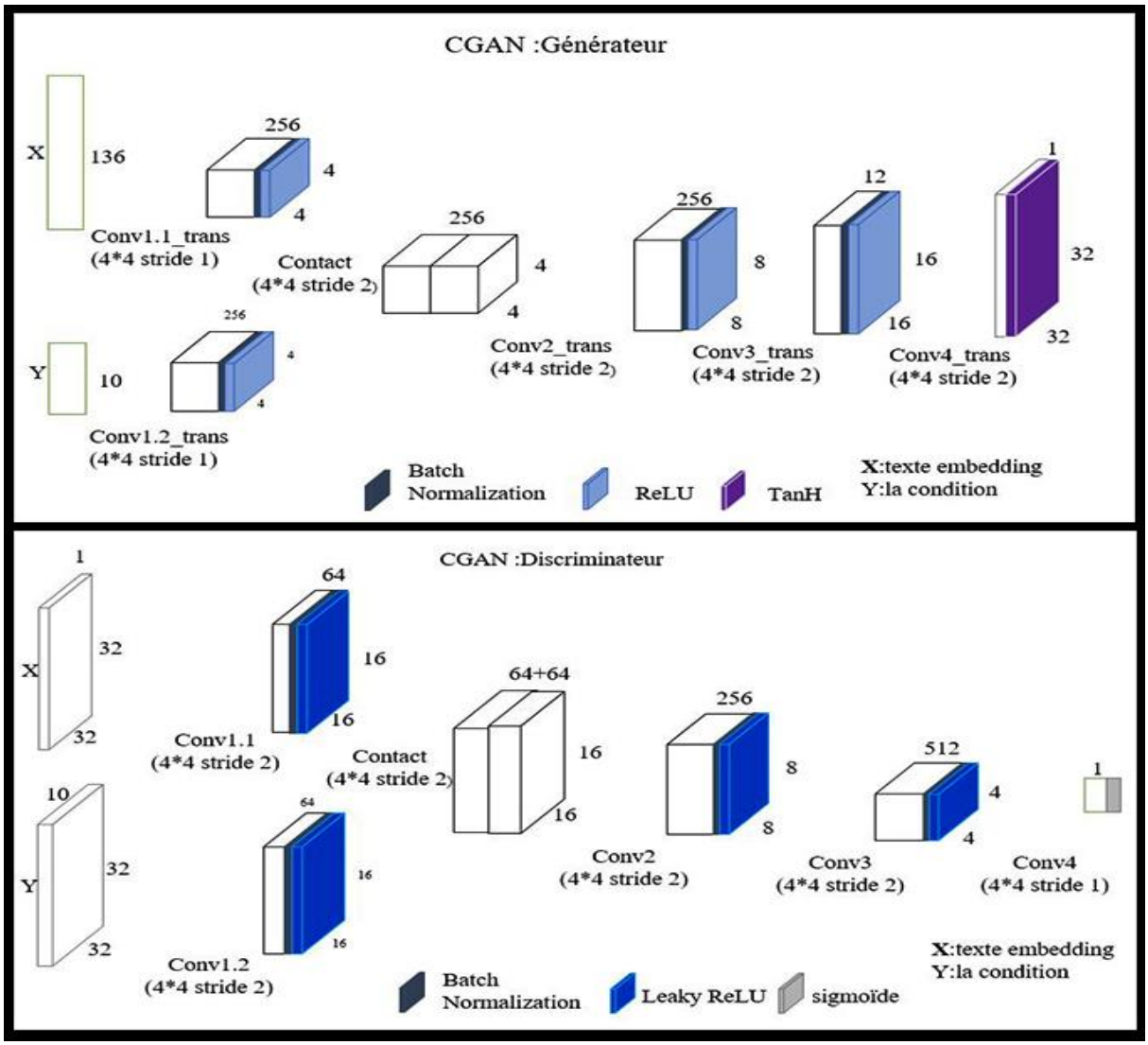


Figure 3- 8:schéma représente l'architecture de CGAN .

5.2.2. Architecture de génération de formes CWGAN

❖ **Le générateur :**

Notre générateur est similaire à celui donné précédemment. Toutefois, nous ajoutons le label y est en utilisant une concaténation. L'architecture à implémenter et illustrée sur **« la figure (3-9) »** est détaillée ci-après :

L'entrée de réseau a deux branches, la première branche est la description textuelle (x) qui est encodée avec une taille de 136 (texte embedding), elle est reliée à un vecteur de bruit z de taille 8. La deuxième branche est la condition (y) dans cette description, elle représente les fonctionnalités nécessaires que nous voulons voir dans la forme 3D générée (la couleur est la catégorie ...).

Les (X et Y) traversent d'abord une couche entièrement connectée (de fully connected) de 32 768 neurones, puis effectuent une normalisation par lots (batch normalisation), où la fonction d'activation utilisée est Relu. Une fois le réchappe de fully connected complètes, des 512 features map de taille 4*4 sera créée.

Ces features map passé par 4 couches de convolution transposée respectivement avec des tailles de (512*4*4), (256*8*8), (128*16*16), (4*32*32), suivie d'une couche de batch normalisation et d'une fonction d'activation Relue.

Cette fonction force les neurones à retourner des valeurs positives sauf le dernier qui a une fonction d'activation Sigmoïde .La sortie du générateur sera un ensemble de voxels de (x; y; z; couleur) = (32; 32; 32; 4).

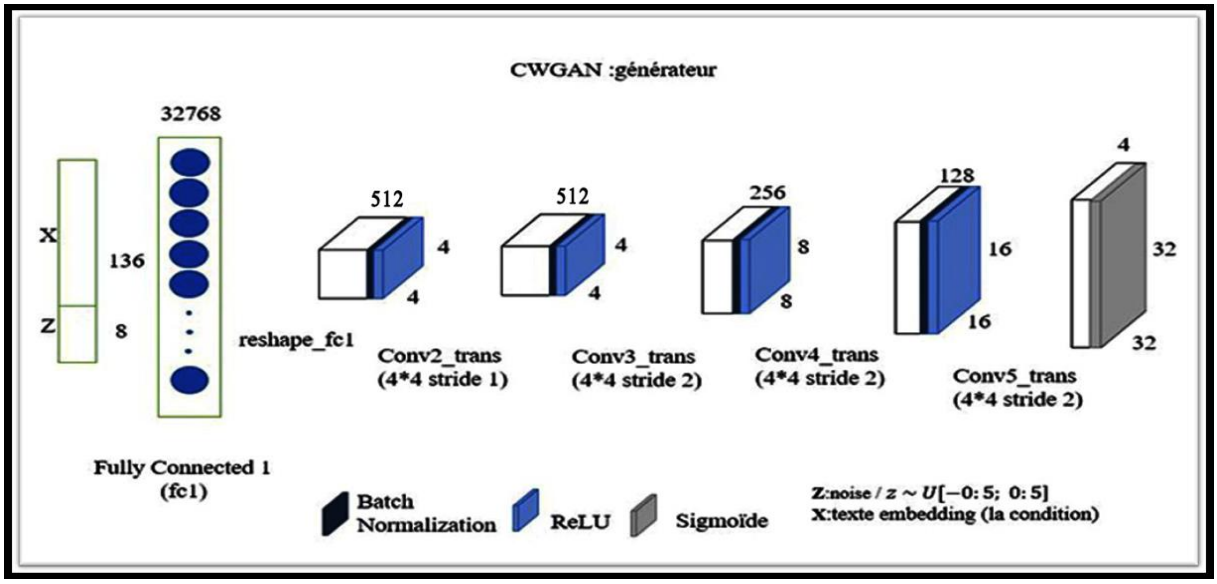


Figure 3- 9:montre l'architecture détaillé de générateur de CWGAN .

Le discriminateur : Les voxels 3D générés (32 ; 32 ; 32 ; 4) entrent dans le Critique et se transforment en un vecteur unidimensionnel via des couches convolutives (Conv1, Conv2, Conv3, Conv4 et Conv5) de taille 256 filtres. nous utilisons une fonction d'activation Leaky Relu qui fuit avec un leak raté de 0,2 dans toutes les couches.

L'incorporation de texte (texte embedding) (X) passe à travers deux couches fully connected ont chacune 256 neurones pour réduire la dimension de l'incorporation de description de texte où la fonction d'activation utilisée est le Leaky Relue et une batch normalisation.

Nous concaténons les deux vecteurs (de forme et de texte) qui sont représentés comme la couche de concaténation (contacte), pour apprendre conjointement les caractéristiques de la

forme et de texte qu'il doit été jugé et les mettre comme entrée d'un réseau de neurones composé de trois couches fully connected de taille (128, 64, 1) neurones où la fonction d'activation utilisée est Leaky relu sauf la dernière qui a une activation sigmoïde.

Pour le critique Wasserstein GAN, nous n'utilisons pas la normalisation par lots (la batch normalisation) car celle-ci est utilisée dans les implémentations de non Wasserstein GAN. **« voir figure 3-10 »**

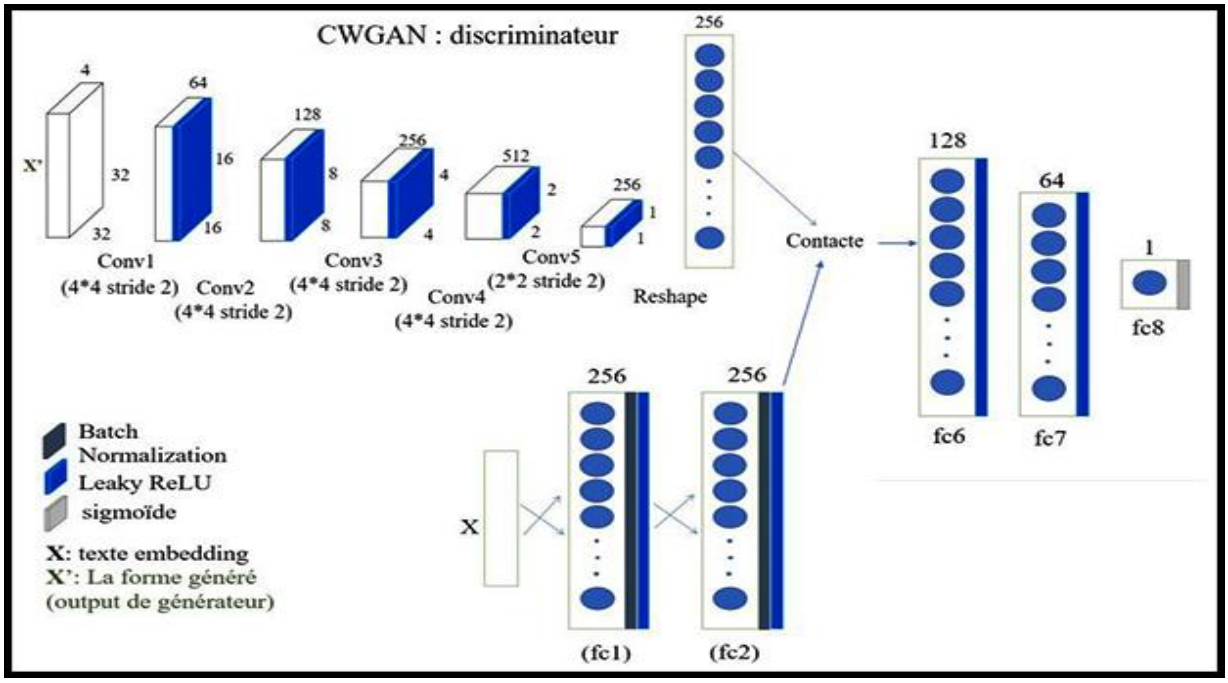


Figure 3- 10:montre l'architecture détaillé de Discriminateur de CWGAN .

5.3.L'entrée de modèle (Input model)

La manière la plus directe pour former un GAN conditionnel est de traiter les formes et l'incorporation de descriptions comme des échantillons conjoints, et de juger si l'ensemble "forme générée + texte" est vrai ou faux en observant le discriminateur.

Cependant, cette méthode est un peu naïve : elle ne fournit pas au discriminateur des informations sur le fait que l'image est correctement générée selon la description lors de l'apprentissage.

Tel que dans l'autre cas de CWGAN, le critique sera jugé pour observer deux entrées différentes :

- **Forme correcte (réel) :** avec le texte correct (description de texte correspondant)
- **Forme générée (fake) :** avec un texte aléatoire.

On aura deux types d'erreurs et cela pourrait compliquer la dynamique d'apprentissage, c'est pourquoi, il est nécessaire d'enregistrer séparément ces deux sources d'erreurs différentes:

- **Des formes réelles :** mais des erreurs de correspondance d'informations conditionnelles (Real match) .
- **Des formes irréelles :** avec n'importe quel texte **(Fake match).**

En plus de l'entrée réelle / fausse, l'auteur a ajouté une troisième entrée .

- **Forme réelle :** avec un texte erroné (ne correspond pas aux formes) **(Real mismatch).**

Et le discriminateur doit être capable de distinguer ce type d'erreur et à marquer comme fake.

Alors l'objectif de notre modèle CWGAN est :

$$\mathcal{L}_{CWGAN} = E_{t\sim pT}\big[D\big(t, G(t)\big)\big] + E_{(\tilde{t},\tilde{s})\sim p_{mis}}[D(\tilde{t},\tilde{s})] - 2E_{(\hat{t},\hat{s})\sim p_{mat}}[D(\hat{t},\hat{s})] + \lambda_{GP}\mathcal{L}_{GP} \quad (14)$$ **[58]**

$$\mathcal{L}_{GP} = E_{(\bar{t},\bar{s})\sim p_{GP}}[(\|\nabla_{\bar{t}} D(\bar{t},\bar{s})\|_2 - 1)^2 + (\|\nabla_{\bar{s}} D(\bar{t},\bar{s})\|_2 - 1)^2]$$

Où D est le critique, G est le générateur, et $pmat$ et $pmis$ correspondent respectivement aux paires de forme et de texte correspondent (**Matching text_shape**) et de forme de texte qui ne correspondent pas (**Mismatching text_shape**), et t c'est le texte embedding concaténées avec le vecteur de bruit, s est le voxel 3D, et $p\tau$ est la distribution de probabilité de l'incorporation de texte.

Notez que nous additions les gradients pour toutes les variables d'entrée de D pour faire la pénalité de gradient (GP).

❖ **L'objectif de G :** est de créer des échantillons qui trompent D, cela minimiser la probabilité que la forme ne provienne pas de vraie distribution de données.

❖ **L'objectif de D :** est de maximiser la probabilité que la condition de forme sur le texte provienne de la vraie distribution de données et de minimiser la probabilité de la forme qui ne provienne pas de la vraie distribution de données.

Conclusion

Notre chapitre a été divisé en deux partie importantes, tel qu'on a compris:
à partir de la première partie, qu'avant d'utiliser la description textuelle elle doit être vectorisée, puis on doit appliquer le modèle CNN_RNN pour construire les incorporations de texte (texte embedding), et ainsi, la matrice de voxel doit passer sur le modèle 3D CNN pour construire les incorporations de forme (Shape embedding)

Et enfin, on a pu saisir que la jointure des deux méthodes d'apprentissage par association LBA et apprentissage métrique ML est dédiée pour la récupération des formes à partir de texte.

À partir de la deuxième partie, l'utilisation de l'architecture CWGAN et la méthode d'arithmétique vecteur a aidés le modèle pour générer des nouvelles formes 3D colorées .

Pour la partie qui suit on va entamer notre quatrième chapitre qui est le dernier de notre mémoire.

IV.Chapitre

l'implémentation de modèle et la discussion des résultats obtenue

1. La méthodologie de la recherche.
2. Expérimentation et évaluation des résultats obtenue.

IV. Chapitre 4: L'implémentation et la discussion des résultats obtenue.

Introduction

Notre dernier chapitre est divisé en deux parties essentielles, tel que dans :

premièrement on va voir la méthode de notre travail, dans cette dernière on va expliquer les outils de recherche qu'on a utilisés durant la période de réalisation de notre projet comme :

les bibliothèques (tensorflow ...) environnement (google collab ...) les datasets (shapeNet et primitive) qui ont voxèlisè et augmenté par des descriptions de texte .

Deuxièmement on va montrer les différents résultats obtenus en matière de précision et d'erreur, et comparer entre eux et les justifier. À la fin on va vérifier la validité de notre hypothèse.

1. La méthodologie de la recherche

1.1.L'environnement de travail

A) Google Colaboratory

Google Colaboratory **[68]** ou bien le colab est un service cloud gratuit offert par google composer d'un bloc note Jupiter. C'est une plateforme dédiée pour :

- La formation et la recherche dans l'apprentissage automatique.
- L'entrainement des modèles de machine Learning directement dans le cloud sans aucune d'installation sur ordinateur.

Plus important encore, Colab prend en charge de nombreuses bibliothèques d'apprentissage automatique populaires qui peuvent être facilement chargées dans votre ordinateur portable et une Accélération GPU gratuite.

B) Bloc note jupyter

Concernant le bloc note jupyter **[68]** :est une application web Open Source qui n'a pas besoin d'avoir une connexion Internet pour l'utiliser permettant de :

- Il crée et partage des documents contenant du code, des équations, des images et du texte.
- Effectuer le traitement de données et la modélisation statistique.
- La visualisation de données de machine Learning .
- Jupyter est disponible par défaut dans la distribution anaconda.

C) Python

Python est un langage de programmation **[69]** open source parmi les plus langages employés par les informaticiens, ce dernier il est dédié pour :

- La gestion d'infrastructure et l'analyse de données dans le domaine du développement de logiciels .
- Le python offert aux développeurs une exécution des codes plus rapidement par rapport aux autres langages, sauf les débutants prend un petit de temps pour prise à la main .

Les principales utilisations de Python par les développeurs sont :

- la programmation d'applications.
- la création de services web.
- la génération des codes .
- la métaprogrammation.

Techniquement, ce langage servira surtout pour le Scripting et l'automatisation (interaction avec les navigateurs web). Le python est disponible maintenant en deux versions 2 et 3.

1.2.Liste des bibliothèques utilisé

A) TensorFlow

Tensorflow **[70]** est une bibliothèque rendue open source par Google. Elle a subi plus de 21000 modifications par la communauté, et est passé en version 1.0. Tensorflow est dédiée pour la machine Learning. Il s'agit d'une boîte à outils permettant de résoudre des problèmes mathématiques extrêmement complexes, Tensorflow offre au chercheur de développer des architectures d'apprentissage expérimentaux et de les transformer en logiciels. Il regroupe un grand nombre de modèles et d'algorithmes de machine Learning et de Deep Learning.

Cette bibliothèque permet notamment :

S'entraîner et d'exécuter des réseaux de neurones pour la classification de chiffres manuscrits.

La reconnaissance d'image, les plongements de mots, la traduction automatique, ou encore le traitement vidéo.

Les applications de TensorFlow peuvent être exécutées sur une machine locale, cluster sur le Cloud, des appareils mobiles ios ou Android, ou même des CPU et des GPU.

Par l'utiliser à google cloud platform, il est possible d'exécuter TensorFlow sur la TPU (TensorFlow Processing Unit) pour profiter d'une importante accélération.

Il y a plusieurs types de Tensorflow comme :

- (Tensorflow-GPU, Tensorflow-IO, Tensorflow-addon…)

B) Keras

Est une bibliothèque **[71]**Open source écrit en python elle permet d'interagir avec les algorithmes de réseaux de neurones profonds et de la machine Learning . Keras est capable de fonctionner sur Tensorflow, theano, PlaidML et autres .keras et commencer dans le cadre d'un projet de recherche pour le system d'exploitation intellige neuro-électronique. La plateforme de keras offre également une prise en charge des réseaux de neurones récurrents et convolutionnels.la bibliothèque a 22% d'utilisation parmi ses plus de 200000 utilisateurs.

C) Pynrrd

Est un module pur python **[72]** dédier pour lire et écrire des fichiers NRRD dans des tableaux numpy.et aussi est une bibliothèque et un format de fichier pour :

- La représentation et le traitement de données.
- Soutenir les applications de visualisation scientifiques et le traitement d'images .

D) Fichier nrrd

Ce document définit le format de fichier Nrrd **[73]** en plus cherche également à fournir une justification telle que les fichiers NRRD doivent se terminer par «nrrd » .

Les fonctions nrrdRead () / nrrdLoad () et nrrdWrite () / nrrdSave () de la bibliothèque nrrd sont censées prendre en charge complètement le format décrit ici, mais pour l'instant il n'y a pas de suite de tests.

E) Numpy

Numpy **[74]** est une bibliothèque logicielle libre et open source et aussi et une extension de langage de programmation python, elle est dédiée pour manipuler les matrices ou bien les tableaux multidimensionnels et leurs fonctions mathématiques et permet de :

Il crée une table directement à partir d'un fichier.

- Sauvegarder un tableau dans un fichier .
- Manipuler les vecteurs , polynômes .

NumPy est la base de SciPy, regroupement de bibliothèques Python autour du calcul scientifique.

F) EasyDict

Permets d'accéder **[75]**aux valeurs de dict en tant que attributs (fonctionne récursivement). Une notation par point de propriétés du type Javascript pour les textes python.

G) pyymal

Pyyaml **[76]** est un analyseur et un émetteur YMAL pour python tel que :
YAML est un format de sérialisation de données conçu pour :

- La lisibilité humaine .
- L'interaction avec les langages de script.
- Les fonctionnalités de pyyaml.
- Un analyseur complet YAML .
- Pyyaml peut analyser tous les exemples de la spécification.

L'algorithme d'analyse est assez simple pour être une référence pour les implémenteurs d'analyseurs YAML.

1.3.La configuration des PC

	PC_1_HP	**PC_2_DELL**
Processeur	Intel(R) core(TM) i7-5600U CPU@ 2.60 GHz 2.60 GHz	Intel(R) core(TM) i5-5200 CPU@ 2.20 GHz 2.20 GHz
Mémoire RAM	8,00 GO	4,00 GO
Type de system	System d'exploitation 64 bits, processeur x64	System d'exploitation 64 bits, processeur x64
Style de fonction tactile	La fonctionnalité d'entrée tactile ou avec un stylet n'est pas disponible sur cet écran	

Tableau 4- 1:représente la configuration de pc.

1.4.La méthode d'installation des bibliothèques

La bibliothèques	La méthode d'installation
Tensorflow	! pip install tensorflow .
Tensorflow-GPU	! pip install tensorflow-gpu .
Tensorflow_io	! pip install tensorflow_io .
Tensorflow_addon	! pip install tensorflow-addons .
Keras	! pip install Keras
Pynrrd	! pip install pynrrd.
Numpy	! pip install numpy .
Easy_Direct	! pip install numpy .
pyymal	! pip install PyYAML .

Tableau 4- 2 :représente les commandes d'installation des bibliothèques .

1.5.Les jeux de données, la voxélisation et prétraitement de text.

1.5.1. Présentation de Shape Net

Un projet de machine Learning commence généralement par un problème à résoudre et par un jeu de données comme une solution de ce problème.

Une forme peut être représentée comme une grille de voxel 3D coloré RGB. Plusieurs dataset existent dans la littérature, toutefois on s'est focalisé le dataset Shape Net **[77].**

ShapeNet **[77]** fournit une vue des données contenues dans une catégorisation hiérarchique selon les synsets WordNet **[78].**Contrairement à d'autres référentiels de modèles, ShapeNet fournit également un riche ensemble d'annotations pour chaque forme et des correspondances entre les formes. Les annotations comprennent des attributs géométriques tels que les vecteurs d'orientation verticale et frontale, les pièces et les points-clés…etc., ces attributs fournissent des ressources précieuses pour le traitement, la compréhension et la visualisation des formes 3D d'une manière consciente de la sémantique de la forme. Cet ensemble de données, on s'intéresse

aux différentes catégories de tables et de chaises car elles contiennent de nombreuses instances avec des variations d'attribut dans la géométrie, la couleur et le matériau. **«Voir la figure 4-1 »**

On a augmenté et enrichi notre dataset de Shape Net par des descriptions en langage naturel (75 344 descriptions en langage naturel) on collecte des descriptions en langage naturel fourni par des personnes sur la plateforme de crowdsourcing Amazon mechanical Turk ***(voir l'annexe N°3),*** en moyenne de 5 descriptions pour chacune des 15000 chaises et tables, on a produit également des voxèlisations en couleur des maillages 3D ***(voir l'annexe N°2)*** CAD en utilisant une surface hybride basée sur une vue et une surface méthode d'échantillonnage, suivie d'un sous-échantillonnage avec un filtre passe-bas dans l'espace voxel.

Alors ont représenté notre ensemble de données d'une forme de paires (x; s), tel que :

- $\boldsymbol{x}$:Indique la description textuelle qu'est représentée comme un vecteur de longueur arbitraire d'indices de mots. Chaque indice correspond à un mot particulier du vocabulaire.
- $\boldsymbol{s}$: Représente Grille de voxel de forme 3D ou $S \in R^{v^{3 \cdot c}}$ tell que v est le nombre maximal de voxels le long de n'importe quelle dimension et c est le nombre de canaux par voxel (4 pour RGB + occupation).

Figure 4- 1:figure représente le jeu de données Shape Net .**[79]**

1.5.2. Jeu de donnée primitive

Pour la génération des formes primitives selon l'auteur Chen **[58]** on a créé un ensemble de données de primitives géomatiques 3D avec des descriptions textuelles correspondantes. On a choisi 6 types de primitives (cuboïdes, ellipsoïdes, cylindres, cônes, pyramides et tores) et 14 variations de couleur et 9 variations de taille. Cette variation de couleur et taille est soumise à

des perturbations aléatoires générant 10 échantillons de chaque 756 de chaque 756 configurations primitives possibles, soit au total, 7560 formes voxèlisèes.

Les perturbations sont obtenues à l'aide d'une application d'un bruit échantillonné uniformément pour la couleur de la forme dans un espace HSV tel que ;

- % 5 de bruit est pour la teinte.
- 10% pour la saturation et la valeur de couleur.
- 15% de bruit pour les échelles de hauteur et de rayon de la forme.

Cet exemple montre un objet 3D voxel coloré de primitive avec trois descriptions textuelles.

« voir la figure 4-2 »

Lors de la création des modèles de phrase pour l'ensemble de données synthétique, ils ont créé des descriptions textuelles correspondent avec une approche basée sur un modèle qui contient des attributs pour : « la forme », « la taille », « la couleur » dans plusieurs niveaux.

Toutes les descriptions textuelles sont présentées comme dans l'exemple suivant :

« À large red cylinder is narrow and tall » « un grand cylindre rouge est étroit et haut »

Ils ont généré 191 850 descriptions en total c'est-à-dire une moyenne d'environ 255 descriptions par configuration.

*Figure 4- 2:un objet 3D voxèlisè de primitive avec ses trois descriptions textuelles.***[58]**

1.6.Voxelisation

Le terme voxélisation décrit la conversion de tous types d'objet géométrique ou volumétrique comme la courbe, la surface, les solides ou les données tomographiques calculées en données volumétriques stockées dans un tableau 3D de voxels **[80]**,L'idée de base de l'algorithme de voxélisation est d'examiner si les voxels appartiennent ou non à l'objet d'intérêt et d'attribuer une valeur de 1 ou 0, Qu'est-ce qu'une grille de voxels **[81]**

Un voxel (a) représente une valeur sur une grille régulière (b) dans un espace tridimensionnel **(voir la figure 4-3)**. Comme pour les pixels dans une bitmap 2 D, les voxels eux-mêmes n'ont généralement pas leur position (coordonnées) explicitement codée avec leur coordonnée.

Les voxels sont fréquemment utilisés dans la visualisation et l'analyse de données médicales et scientifiques (par exemple SIG).

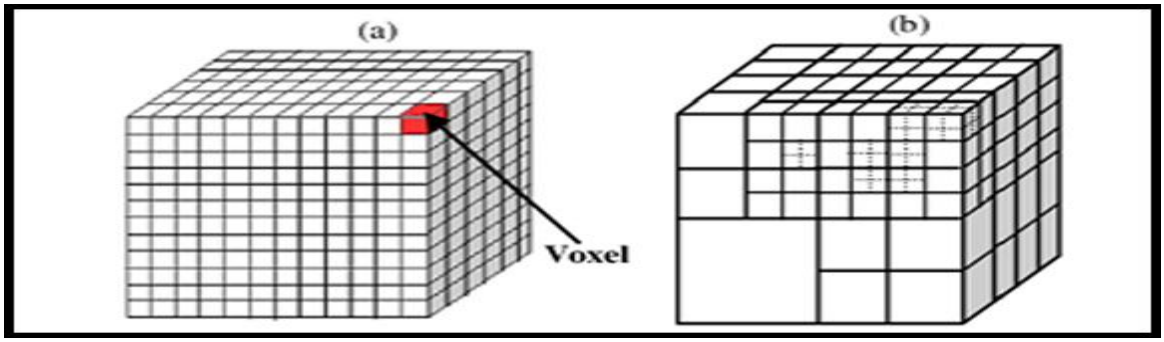


Figure 4- 3:schéma représente les grilles de voxel.**[81]**

1.7.Prétraitement des descriptions en langage naturel

Par rapport au prétraitement des descriptions textuelles en langage naturel, le texte est en minuscules, tokenisé ***(voir l'annexe N°4)*** et lemmatisé ***(voir l'annexe N°5)*** à l'aide du pipeline Spacey ***(voir l'annexe N°6)***.Le Language Tool2 ***(voir l'annexe N°7, N°8).*** pour corriger les fautes d'orthographe de l'ensemble de description en langage naturel de Shape Net. On a traité les descriptions textuelles et rejeté environ 16 descriptions à cause de leur longueur.

Les termes en basse fréquence (survenant ≤ 2 fois) ont été mappés sur le jeton UNK (token UNK) ***(voir l'annexe N°9)*** qui est une représentation de dictionnaire a été établir ce dernier par l'utilisation des termes restants.

- Shape Net possède en total 8147 mots uniques. Après avoir supprimé les mots à basse fréquence (mot vide) et les avoir réduits dans le jeton UNK, à la fin, on retrouve une taille de vocabulaire de 3588 mots.
- Primitive ne possède pas des mots à basse fréquence (avec le jeton UNK).

Le tableau ci-dessous présent les statiques de deux jeux de données. Le jeu de données primitives a beaucoup plus de description par forme, mais a une taille de vocabulaire beaucoup plus petite et ses descriptions sont plus courtes. **« Voir la figure 4-4 ».**

Table 4: Dataset statistics.

	Primitives	ShapeNet
Shapes	7,560	15,038
Descriptions	191,850	75,344
Descriptions per shape	255	5
Total words	1.3M	1.2M
Ave words/description	6.7	16.3
Max words/description	8	96
Unique words	76	8147
Vocabulary size	77	3,588

Figure 4- 4:montre les statistiques de deux bases de données .**[58]**

1.8.Les étapes de traitement des données

Dans cette partie on va expliquer : les étapes de traitement de notre base de jeu de données (dataset). Et montrer les changements qu'on a attribut sur notre architecture en expliquant les avantages et les inconvénients de chaque changement.

Ensuite on va évaluer notre résultat, on utilise les métriques d'évaluation et les justifier, et afficher des exemples sur les formes générer de bonne et mauvaise qualité dans les deux cas de changement. On conclura par la vérification des hypothèses qu'on a proposées.

Pour rappel, le traitement des données est composé de trois phases:

Phase d'entrainement : c'est la partie responsable sur l'affichage d'un ensemble (de epoch et steps et son temps d'entrainement ....) en plus l'échantillon de données utilisées pour ajuster le modèle.

Phase de validation : c'est la partie responsable pour estimer l'erreur de prédiction pour la sélection du modèle sur l'affichage de (précision et le rappel, accuracy...) à la fin de chaque validation, il y a une restauration de dernier point de contrôle (checkpoint) et un **« *model-ckpt* »** est créé dans le dossier d'entrainement. Ce modèle est composé de trois fichiers (data, index, méta).

Phase de test : est utilisé pour évaluer l'erreur de généralisation du modèle final choisi. Généralement, dans cette partie, on utilise le dernier ***« model-ckpt.data »*** générer à partir de l'étape précédente, et à la fin de test il y a des fichiers créés à l'intérieur de dossier test pour confirmer le bon fonctionnement de traitement de données.

Pour arriver à ces trois phases il existe d'autres paramètres importants programmés au niveau de chaque commande d'entrainement effectué, comme :

Le nombre d'époques (epoch) :décrit le nombre de fois où l'algorithme voit l'ensemble des données. Lorsque l'algorithme a vu tous son échantillon de l'ensemble de données, une époque s'est terminée.

Le taux d'apprentissage (Learning rate) :est un hyperparamètre configurable utilisé dans l'apprentissage des réseaux de neurones. C'est une petite valeur comprise entre 0,0 et 1,0. Son objectif est de contrôler la rapidité avec laquelle le modèle est adapté au problème.

La normalisation par lot (batch size) :c'est un paramètre très important dans cette partie. Ce dernier fait référence au nombre d'exemples d'entraînement utilisés dans une itération. C'est-à-dire celui qui augmente ou bien minimisé le nombre d'epoch et steps.

Lorsqu'une étape d'entrainement (step) est terminée, on affiche certaines informations comme :

La perte (loss) :est un indicateur d'une mauvaise prédiction. C'est-à-dire le nombre qui indique à quel point la prédiction de modèle était mauvaise sur un seul exemple .si la valeur de perte est nulle, la prédiction du modèle est parfaite sinon la perte est plus importante.

Data fetch (sec/steps) : c'est le temps pris par le step pour récupérer les données.

Queue size : c'est la taille de la file d'attente.

Train step (sec/step) : le temps par seconde pour entrainer un step.

La précision (Accuracy) : d'un algorithme de classification par apprentissage automatique est un moyen de mesurer la fréquence à laquelle l'algorithme classe correctement un point de données

« La figure 4-5 » montre un exemple sur l'affiche d'un step durant la phase d'entrainement.

```
---------- train step 018000 ---------------
INFO:tensorflow:2020-07-19 19:23:59.570936      loss: 6.676026532659307e-05
INFO:tensorflow:2020-07-19 19:23:59.571133      queue size: 7/20
INFO:tensorflow:2020-07-19 19:23:59.571191      data fetch (sec/step): 0.14
INFO:tensorflow:2020-07-19 19:23:59.571236      train step (sec/step): 0.33
INFO:tensorflow:Running validation.
INFO:tensorflow:Evaluated 1472 samples.
INFO:tensorflow:Accuracy: 0.9932065217391305
INFO:tensorflow:Previous validation accuracies:
INFO:tensorflow:[0.99388587 0.99184783 0.99320652 0.99184783 0.99184783]
INFO:tensorflow:Current validation accuracy: 0.9932065217391305
INFO:tensorflow:Saving checkpoint (step 18000).
```

Figure 4- 5:represente un affichage de step .

Chacun de ces paramètres avoir un rôle important dans l'entrainement, mais cette fois-ci on base beaucoup plus sur le paramètre de normalisation par lot (batch size) .

Généralement les batchs size proposés dans l'entrainement des données soient : 8 ou 32 ou 64 ou 128 ou 254...et au fait que chacun d'eux possède ses propres avantages et inconvénients, on ne peut pas dire quels sont le bon et l'idéal d'après l'ensemble proposé, mais ! à travers la comparaison des résultats obtenus aux fins du traitement on peut connaitre la valeur du paramètre la plus adéquate pour notre modèle .

1.9.Changement de paramètre

Pour affirmer la validité à tout ce qu'on a dit, dans la partie précédente, on prend ce cas et on va expliquer les changements qu'on a effectués durant notre entrainement de modèle tel que :

Lorsque nous sommes arrivés à l'entraînement de notre architecture CWGAN, on a lancé ce dernier pour la première fois avec le batch size 8 on a remarqué que l'entrainement il marche bien, mais malheureusement le nombre d'epoch est accès grand environ de 092300 et le CWGAN prend un large de temps pour entrainer et afficher une epoch en plus on ne peut pas terminer l'entrainement dans un seul jour au minimum prend une période de 25 jours

(Le temps d'exécution est trop couteux. Ceci revient a la grande dimension de la base ce qui nécessite l'utilisation d'un GPU au lieu d'un CPU).

et tanque le batch size c'est le paramètre responsable sur l'augmentation et la minimisation de nombre d'epoch et step on a essayé d'entrainer le modèle avec le batch size 32 on a trouvé qu'avec ce dernier le nombre d'epoch est devenu la moitié de premier , c'est-à-dire environ de 023200 steps au minimum prend une période de 10 jours **« Voir le tableau 4-3 »**, et comme on a mentionné déjà qu'on ne peut pas savoir qu'elle est le meilleur dans notre cas. En plus on ne peut pas risquer de choisir l'un parmi les deux avant les tester . C'est-à-dire ont été obliger de lancer l'entrainement en fur mesure, l'un avec le batch size 8 et l'autre avec le batch size 32 en deux PC diffèrent .

Batch size	8	32
Nombre d'epoch par steps	Très grand environ de 092300	Petit environ de 023200
Nombre d'échantillons entrainer par epoch	Un lot de 8	Un lot de 32
Temp d'afficher steps par epoch	5 minutes	5 jusque 8 minute
Période de fin d'entrainement	25 jours	10 jours

Tableau 4- 3:tableau représente la comparaison entre les deux valeurs de batch size .

1.10. Les avantages et les inconvénients

- **Avantage de batch size 8**

Il prend un petit nombre d'échantillons dans son lot pour les identifier en fonction de chaque époque, en plus il affiche le dernier après quelques secondes (c'est-à-dire que cela ne prend pas beaucoup de temps) ce qui permet au réseau d'améliorer l'apprentissage sur les formes présentées dans la base de données.

- **Inconvénients de batch size 8**

Le nombre d'epoch par steps a entrainé plus grand, on remarque qu'à chaque fois le batch size est petit le nombre d'epoch est grand en plus la période d'entrainement très longe il ne nous convient pas dans un projet limiter par le temps.

- **Avantage de batch size 32**

Le nombre d'epoch par steps a entrainé plus petit , on remarque qu'à chaque fois le nombre de steps est grand le nombre d'epoch est petit en plus la période d'entrainement très petite, il nous convient dans le cas où le projet limité par le temps .

- **Inconvénients de batch size 32**

Il prend un nombre grand d'échantillons dans son lot pour les identifier par chaque epoch en plus il affiche cette dernière après quelques minutes c'est-à-dire (il prend un temps) ce qui permet au réseau de soit mieux ou non, pour s'entrainer sur toutes les formes de la base de jeu de données.

2. Expérimentation et évaluation des résultats obtenue

2.1.Métriques d'évaluation

Dans cette étape, nous évaluerons les résultats de notre travail car nous avons constaté que l'idée de visualiser le travail est une partie importante du projet. Telle que Nous essayons de classifier les Shape où les forme (les tablent et les chaises) de notre data set Shape Net (test set) en deux classes :

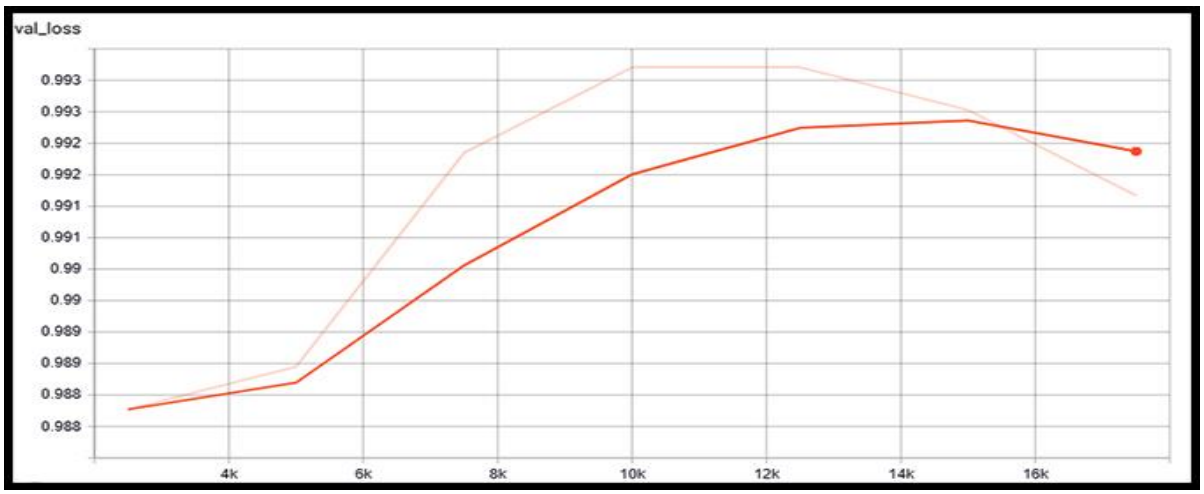
1. Table (11921 tables)
2. Chaise (1486 chaises)

Après avoir alimenté notre modèle par le jeu de données de test, les classifications du système pour chaque forme sont résumées dans la grille ci-dessous. Ou notre système généralement est « précis à 99% ». Mais il y a quelques différences entre les deux batch size :

Pour le batch size de 8 : précis à 99.1% .

```
INFO:tensorflow:Accuracy: 0.9911684782608695
INFO:tensorflow:Previous validation accuracies:
INFO:tensorflow:[0.99252717 0.99320652 0.99320652 0.99184783 0.98845109]
INFO:tensorflow:Current validation accuracy: 0.9911684782608695
```

Figure 4- 6:represente résultat de précision batch size 8 .

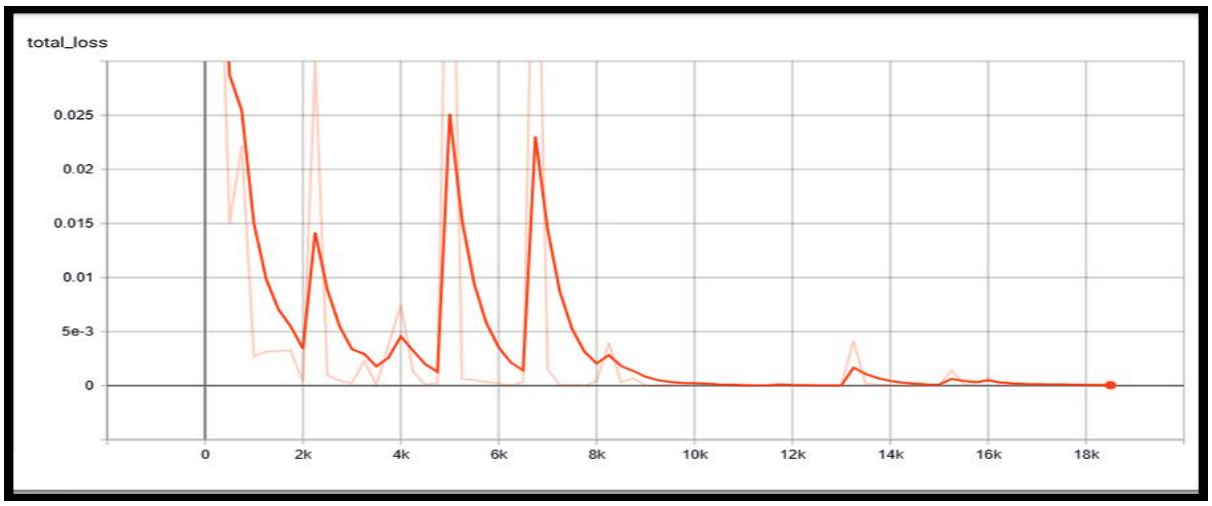


Graphe 4-1:montre le graphe de résultat de la de précision batch size 8 .

Le **« graphe 4-1 »** montre le graphique de résultat de la précision que nous avons obtenu d'après le traitement de l'ensemble de jeux de données (dataset) tel qu'on remarque qu'il y a une augmentation permanente avec le nombre de steps, ceci reflète qu'à chaque step le modèle ap-

prend plus d'informations et si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre des steps et vice-versa.

Le taux de précision de batch size 8 est de 99,1%.



Graphe 4-2:montre le graphe le résultat de la perte de modèle en batch size 8.

Le **« graphe 4-2 »** montre le graphique de résultat de la perte que nous avons obtenu d'après le traitement de l'ensemble de jeux de données (dataset) tel qu'on remarque qu'il y a une démunissions permanente avec le nombre de steps c'est-à-dire la perte de classification de notre modèle pour les formes est environ de 0 par l'utilisation de paramètre de batch size 8.

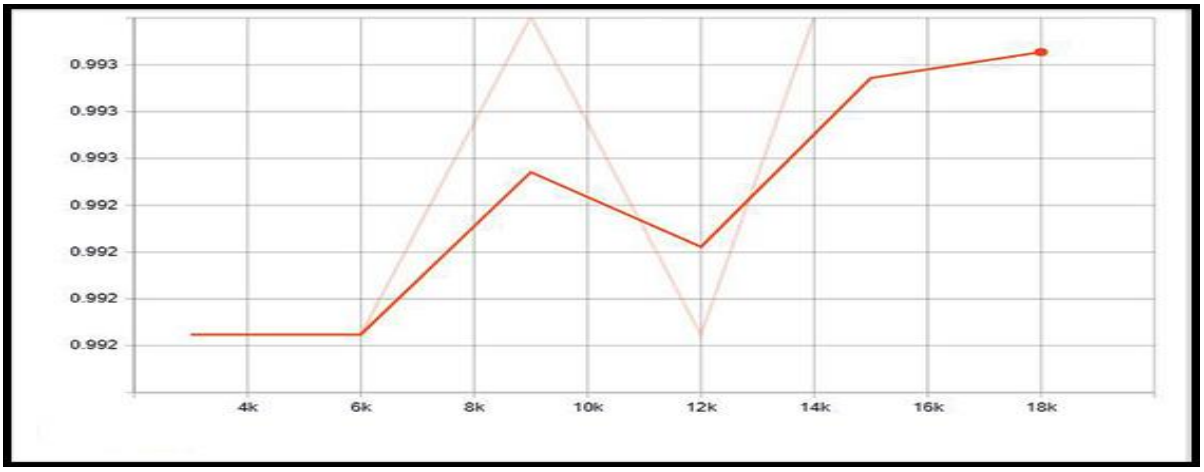
Le batch size de 32 : précis à 99.3%

```
INFO:tensorflow:Running validation.
INFO:tensorflow:Evaluated 1472 samples.
INFO:tensorflow:Accuracy: 0.9932065217391305
INFO:tensorflow:Previous validation accuracies:
INFO:tensorflow:[0.99388587 0.99184783 0.99320652 0.99184783 0.99184783]
INFO:tensorflow:Current validation accuracy: 0.9932065217391305
```

Figure 4- 7:représente résultat de précision batch size 32.

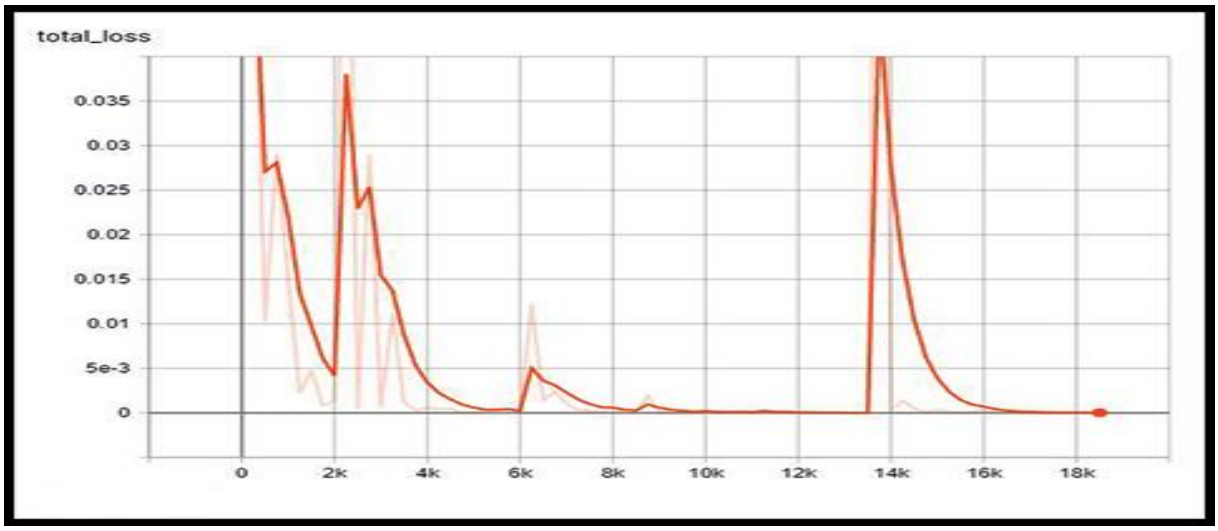
Le **« graphe 4-3 »** montre le graphique de résultat de la précision que nous avons obtenu d'après le traitement de l'ensemble de jeu de données (dataset) tel qu'on remarque qu'il y a une augmentation avec le nombre de steps, ceci reflète qu'à chaque step le modèle apprend plus d'informations et si la précision est diminuée alors on aura besoin de plus d'information pour faire apprendre notre modèle et par conséquent on doit augmenter le nombre des steps et vice-versa.

Le taux de précision de batch size 32 est de 99,3%.



Graphe 4-3: représente résultat de précision batch size 32.

Le **« graphe 4-4 »** montre le graphique de résultat de la perte que nous avons obtenu d'après le traitement de l'ensemble de jeux de données (dataset) tel qu'on remarque qu'il'y à une démunissions permanente c'est-à-dire la perte de classification de notre modèle pour les formes est environ de 0 par l'utilisation de paramètre de batch size 32.



Graphe 4-4:représente résultat de la perte de modèle en batch size 32.

2.1.1. Matrice de confusion

Voici comment calculer la matrice de confusion

Vrai positif

$11921 \longrightarrow 100\%$

$? \longrightarrow 99.1\%$

$$\text{Vrai positif} = \frac{11921 * 99.1}{100} \qquad \text{Vrai positif} = 11813.7$$

Faux négatif

$$faux\ négative = 11921 - 11813.7 = 107.3 \qquad faux\ négative = 107.3$$

Vrai négatif

$1486 \longrightarrow 100\%$

$? \longrightarrow 99.1\%$

$$\text{Vrai nègatif} = \frac{1486 * 99.1}{100} \qquad \text{Vrai nègatif} = 1472.6$$

Faux positif

$$faux\ positif = 1486 - 1472.6 = 13.4 \qquad faux\ positif = 13.4$$

Tel que :

Le nombre de modèles égale =11921 et 1486 .

« Le tableau 4-4» résume tous les résultats obtenus .

		Résultat réel	
Résultat prédit		**Table Positive**	**Chaise (non table) Négative**
	Table Positive	**Vraies positives 11813.7**	**Fausses positives 13.4**
	Chaise (non table) Négative	**Fausses négatives 107.3**	**Vraies négatives 1472.6**

Tableau 4- 4:représente les résultats de la matrice de confusion .

1. Vrai positif : forme prédite table + réellement une table .
2. Faux positif : forme prédite table + réellement une chaise.
3. Vrai négatif : forme prédite chaise + réellement une chaise.
4. Faux négatif : forme prédite chaise + réellement une table .

Comme mentionné précédemment (chapitre I), les paramètres que nous utiliserons sont L'accuracy, précision, Recall et F1 score(f-mesure).

❖ **Accuracy**

Combien de formes le système a-t-il correctement classées (c'est-à-dire les vrais positifs ou les vrais négatifs) parmi toutes les formes ?

$$Accuracy = \frac{11813.7 + 1472.6}{11813.7 + 13.4 + 107.3 + 1472.6} \qquad Accuracy = 0.990 \qquad (1)$$

❖ **Précision**

Combien de ces formes étiquetées par le système comme table sont en fait table ?

$$Précision = \frac{11813.7}{11813.7 + 13.4} \qquad Précision = 0.998 \qquad (2)$$

❖ **Recall (Rappel)**

Parmi toutes les tables de dataset, combien de celles-ci le système a-t-il correctement classé comme table ?

$$Recall = \frac{11813.7}{11813.7 + 107.3} \qquad Recall = 0.990 \qquad (3)$$

❖ **Score_F1 (F_mesure)**

$$f1 = \frac{2 * 0.998 * 0.99}{0.998 + 0.99} \qquad f1 = 0.993 \qquad (4)$$

Sur la base de ce qui précède, nous pouvons dire que le système de classification des formes de Shape Net a un **rappel**, une **précision** et une **accuracy** élevés.

Alors la chance qu'une forme identifiée par le système comme table (ou chaise) soit en fait table (ou chaise) est 99.8% .

2.2.Ensemble des formes de bonne qualité

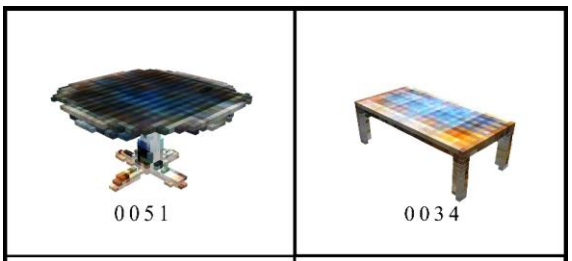
Description textuelle	Batch size 32	Batch size 8
0022 *wood chair with plain backrest and style leg .* ***0154*** *a chair in shape of a heel sandal , the back and seat be of black color and the other part be dark red .*		
0025 *this be a short table , make from wood . it ' brown .* ***0073*** *light brown color rectangle table for use fully wooden material , its use for meeting , club etc. . ,*		
0051 *a queen size bed in gray color with storage space .* ***0034*** *this be a wooden table . it looks like something that would be place outside to hold thing . it also looks like a great storage device . this table be gray colored .*		
0173 *a dark green mahogany wood finish , with green smooth finish on top . a partial glass tops .* ***0136*** *a black rectangular table with two triangular leg .*		

Tableau 4- 5: tableau représente l'ensemble des formes de bonne qualité .

2.2.1. La justification des résultats de Bonne qualité

La qualité des résultats obtenus dans les deux cas indique que le modèle a été entrainer dans des bonnes conditions c'est-à-dire ses paramètres d'entrainement sont bien ajusté en plus la description textuelle présente toutes les caractéristiques de la forme 3D.

(la couleur, catégories...) donc le réseau ne fait pas beaucoup d'efforts pour chercher les mots pertinents parce que tous les paramètres sont présentés d'une façon claire, en plus le réseau a pris suffisamment de temps pour transformer en forme 3D coloré voxèlisè (haute qualité) .

❖ **Remarque**

La condition de CWGAN présenter au niveau du générateur et discriminateur est celle qui détermine les caractéristiques de la forme (couleur, catégorie ....).

2.3.Ensemble des formes de mauvaise qualité

Description textuelle	Batch size 32	Batch size 8
0029 *a gray pool table with six ball arrange close to the middle of the table and right edge . a rack to arrange the ball be center on the left where one may place the ball when start a new game .* ***0072*** *a round table with a white surface that fold down on a hinge in the center to become a semi-circle . the leg be making of brown wood and be braced to a central support .*		
0162 *a red color table make with wood and it has four leg.* ***0161*** *a brown color wooden table with 4 leg . it has white color stripe draw on it in whole rectangle shape .*		
0199 *the traditional kitchen chair , with back support and furnished base , comfortable and light to use around the house .* ***0183*** *a gray chair .it have a high back and a square shape .*		

Tableau 4- 6:tableau représente l'ensemble des formes de mauvaise qualité .

2.3.1. La justification des résultats de mauvaise qualité

La mauvaise qualité des résultats obtenus dans les deux cas indique que le modèle a subi l'un ou tous les problèmes au cours de son entrainement dont :

Les coupures de connexion d'internet au moment que le modèle a commencé de reconnaître les caractéristiques de la forme 3D, comme conséquence certaines formes 3D ,ils avaient des figures incomplètes par exemple (chaise ou bien table sans pied ....).

Dans le cas où le réseau est un grand nombre des formes à entrainer dans son lot à la fois, par exemple 32 ou bien 64 ...

Parfois, le modèle ne peut pas entraîner toute cette taille énorme de formes, c'est pour cela on remarque que certains forme n'était pas bien entraînés .

Dans le cas où les descriptions textuelles sont complexes et contiennent plus d'un mot pertinent c'est-à-dire (mal présenté), et plusieurs caractéristiques dans la forme 3 D, le réseau

peut ne pas être en mesure de les identifier toutes, c'est pour cela on remarque que certains artéfacts sont manquants.

2.4.L'interface graphique de notre application

Les figures **4-8** et **4-9** montrent l'interface graphique de notre système. Nous écrivons d'abord la description textuelle qui contient plusieurs caractéristiques de la forme, puis cliquons sur le bouton « open » . Une fenêtre s'ouvre qui contient la forme 3D avec un panneau de contrôle pour améliorer la qualité de la forme générée.

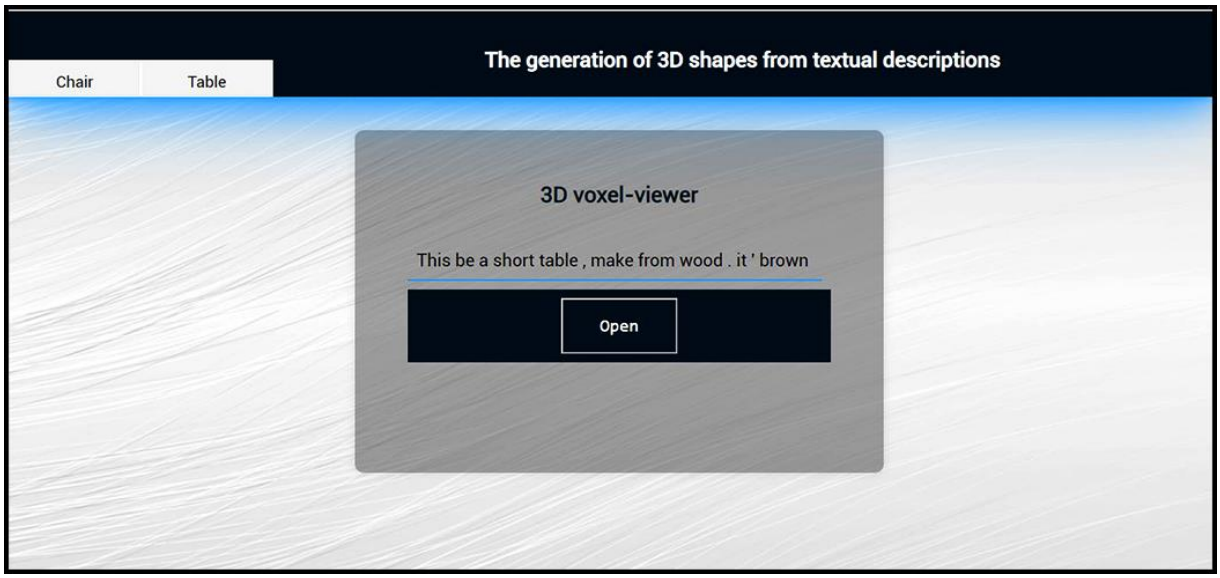


Figure 4- 8:montre le texte saisi en entrée dans l'interface graphique.

Figure 4- 9:montre la forme 3D en sortie dans l'interface graphique.

2.5.La comparaison des résultats avec d'autre approche

Dans cette partie on va discuter sur notre résultat obtenu par rapport aux résultats des autres travaux connexes qui sont en relation avec notre approche .

Comme on a mentionné déjà dans le deuxième chapitre de l'état de l'art , il y avait aucun travail ne précèdent qui traite la génération des formes 3D à partir des descriptions textuelles sauf « *texte2shape*» de l'auteur Chen **[58]**, c'est le résultat de la fusion des deux catégories :

Text en image et image en forme.

Dans notre cas on prend la catégorie qui nous intéresse c'est le troisième texte en forme .

Comme on a dit déjà notre approche a été implémenté par l'utilisation de l'architecture de CWGAN avec les changements de paramètre qu'on a attribués sur cette dernière, *texte2shape* ils ont utilisé de plus de CWGAN une autre architecture de CGAN pour générer les formes 3D. Donc on compare notre résultat obtenu par les résultats obtenus de « *texte2shape »* **[58]**.

Selon les changements de paramètre qu'on a attribué sur l'architecture on a remarqué que notre résultat dans les deux cas, que ce soit le batch size 8 ou bien 32 étaient proches entre eux les uns à des autres . par rapport au batch size 8 on a obtenu une précision de **99.1%** et par rapport au batch size 32 on a obtenu un de **99.3%** .

Cela donne de bons résultats et selon les métriques arithmétiques utilisées, nous avons constaté que notre modèle évaluait bien les formes 3D avec ses descriptions textuelles d'environ 0,99 (99%), ce qui est un bon signe pour nous.

La comparaison de notre modèle avec d'autre modèles :

dans cette partie nous comparons les résultats de notre modèle CWGAN avec les résultats obtenus de CGAN.

Selon l'auteur Chen **[58]** a déclaré que l'utilisation de l'architecture de CGAN donne des résultats de conditionnement et de génération comparables celle de CWGAN, la différence entre les deux c'est qu'au niveau de qualité des formes générer tel que :

L'architecture de CWGAN génère des formes plus réalistes avec des distributions de couleurs plus naturelles et un meilleur conditionnement du texte**.« le tableau 4-5»** montre les formes générer avec ses descriptions textuelles correspondantes.

L'architecture de CGAN aussi réussi de générer des formes 3D correspondant au texte mais ce n'est pas de bonne qualité par rapport au CWGAN tel qu'il reste encore des artefacts structurels et de couleur .

2.6.La vérification des hypothèses proposé

Dans cette section, nous discuterons de la validité des hypothèses que nous avons suggérées précédemment.

H1: on croit que, le changement de paramètre influe sur le nombre d'epoch et le temps d'entrainement et même pour la qualité des résultats obtenus.

H2: on pense que, l'utilisation de la méthode d'incorporation conjointe (joint embedding) de l'apprentissage par association LBA et apprentissage métrique aide le modèle pour récupérer la forme 3D correspondante à une description textuelle saisie en entrée.

H3: selon les travaux de plusieurs auteurs qui ont fait des approches liées à la génération des textes en images, image en forme et texte en forme on suppose que l'utilisation d'une architecture de réseau génératif adversaire GAN avec n'importe quel type c'est une méthode idéale pour garantir la génération des formes 3D.

H4 : on croit qu'utilisation de l'architecture « CWGAN » aide le modèle pour générer une nouvelle forme 3D qui n'existe pas dans l'ensemble de jeux de données.

Pour effectuer les méthodes de recherche ont été besoin plus d'outils comme l'analyse documentaire (articles …) et aussi on va utiliser deux grandes bases de jeu de données (data Set)

Shape Net et primitive contient un ensemble des formes 3D est augmenté avec des descriptions textuelles.

- **Discussion sur H1**

Comme on a expliqué dans la partie de changement de paramètre sur notre modèle et d'après les comparaisons qu'on a effectués sur le batch size 8 et 32 et on a pu d'extraire les avantages et les inconvénients de chacun d'eux, on a confirmé la validité de l'hypothèse H1 car ,effectivement le changement de paramètre influe sur le nombre d'epoch et le temps d'entrainement et même pour la qualité des résultats obtenus.

- **Remarque**

À chaque fois on augmente le paramètre de batch size . le nombre d'époque diminuer et le temps d'entrainement de chaque epoch devient très lord et la qualité des résultats change, donc les valeurs de chaque paramètre, choisi selon la complexité d'architecture.

- **Discussion sur H2**

Dans notre troisième chapitre on a précisé les étapes de conception de notre approche parmi cette dernière on a expliqué deux méthodes importantes lesquelles « l'apprentissage métrique » et « l'apprentissage par associons LBA » tel que :

L'apprentissage métrique permet de relier chaque forme par sa description textuelle la plus adéquate .

L'apprentissage par association LBA : son principe basé sur les associations (groupe) c'est-à-dire elle permet de relier la description textuelle avec toutes ses formes et le cas contraire relier chaque forme avec toutes ses descriptions textuelles .

à travers les principes de deux méthodes, on a compris que la jointure de ces deux méthodes nous a aidées pour récupérer des formes 3D correspondant à ses descriptions textuelles saisies en entrée . Selon les résultats présentés dans les deux **tableaux 4-5 et 4-6** on a confirmé que l'hypothèse H2 elle est vérifiée.

- **Discussion sur H3**

Dans notre deuxième chapitre on a présenté notre état de l'art tel qu'on a collecté les travaux connexes qui sont en relation avec notre approche .on a classé les travaux selon trois catégories puis on a résumé les résultats obtenus par chaque approche dans le **« tableau 2-1 »**, comme vous le savez que les articles choisis dans cette partie dans les trois catégories que ce soit : texte en images, image en forme, ou bien texte en forme. vise à la même objective, c'est la génération (des images ou des formes ...) et selon le **« tableau 2-1 »** on n'a obtenu que la majorité utilisée des architectures de GAN avec des types différents pour arriver à leur objectif.

Et comme on a choisi l'architecture de CWGAN qui était aussi un autre type de GAN donc on comprend que l'hypothèse H3 est vérifiée.

- **Discussion sur H4**

D'après les tests de la commande qu'on a attribué sur notre architecture de CWGAN cette dernière elle donne une sortie d'un ensemble des fichiers contiennent les formes 3 Ds générer, à chaque fois on change les descriptions textuelles on remarque qu'il y a des nouvelles formes 3D générées qui ne rassemblent pas aux précédentes, mais pas de bonne qualité, même s'il existe ce petit problème mais l'hypothèse **H4** restes toujours vérifier.

Conclusion

On conclut notre dernier chapitre par un petit rappel sur tous ce qu'on vu déjà.

À partir de la première partie, nous pouvons saisir que la construction de notre modèle a requis plusieurs outils comme l'équipement physique, l'environnement d'exécution (python et google Colab) et un ensemble de bibliothèque (Tensorflow, Keras…) en plus il est devenu clair que pour la génération de textes en forme 3D colorée on a besoin de passer sur les étapes de traitement de nos données (description textuelle et forme 3 D)pour assurer son bon fonctionnement,

Par rapport à la deuxième partie, elle est consacrée aux expériences que nous avons menées et à la discussion des résultats obtenus, où nous pouvons confirmer les hypothèses que nous avons proposées auparavant :

l'utilisation de l'architecture **CWGAN**, elle nous a aidés pour la génération des formes 3D réalistes à partir des descriptions textuelles et même le changement de paramètres (le nombre des steps, la taille de batch) qu'on a effectué, elle a joué un rôle majeur dans la qualité des résultats obtenus, en influençant sur le nombre d'époques et le temps d'entrainement.

Conclusion générale

À la fin de notre mémoire, nous vous rappellerons les principaux points que nous avons traités précédemment. Cette étude nous a permis de travailler sur un nouveau sujet, qui a été rarement traité auparavant, tel que dans la littérature de la discipline, les auteurs ont accordé une importance particulière à l'application des approches texte en image et image en forme, d'une manière séparée, sans avoir accordé une importance de les appliquer d'une manière combinée, qui donne naissance de notre approche texte en forme.

L'objectif clé de ce travail est divisé en deux objectifs lesquels :

La récupération et la génération des formes 3D qui reflète exactement les mots pertinents d'une description textuelle saisie en entrée. Pour atteindre notre but, il est nécessaire de passer sur une chaine de processus, avant d'utiliser nos données dans n'importe quel modèle, la première des choses à faire est de traiter les descriptions textuelles et les formes 3D pour que la machine devienne capable de les comprendre, tel que :

Le réseau de neurones artificiel comme le convolutionnel récurent (CNN_RNN) est dédié pour construire les incorporations de text (text embedding) .

Le réseau convolutionnel 3D (3D_CNN) est dédié pour construire les incorporations des formes (Shape embedding).

L'apprentissage métrique a comme principe de déterminer la similarité entre les descriptions textuelles (text-to-text) et entre la description et la forme 3D (text-to-Shape) .

L'apprentissage par association a comme principe d'associer sémantiquement les textes.

similaires ensemble et les formes similaires ensemble et de séparer ceux qui sont différents.

La jointure de ces deux méthodes (apprentissage métrique et par association) donne une naissance pour une nouvelle méthode qui est l'incorporant conjointe, cette dernière, elle nous a aidés pour effectuer notre premier objectif qui est la récupération de texte en forme.

Pour assurer la génération des nouvelles formes 3D, on a utilisé l'architecture de CWGAN.

À partir de ce projet, on a amélioré notre connaissance telle qu'on a réussi à créer une application de type machine Learning, basé sur l'intelligence artificielle en plus programmé en langage python et exécuté dans l'environnement de Google Colab .

D'après tout ce qu'on a vu durant ce mémoire, on a compris le cœur de domaine « IA » comme les réseaux de neurones artificiels et les réseaux génératifs GAN avec ses différents types qui nous ont aidé pour générer la forme 3D colorée.

La Conclusion Générale

Afin d’améliorer la qualité de notre travail, on propose de :

Tester notre approche aussi avec les autres fichiers de voxélisation solide 64 et 128 pour améliorer et augmenter la qualité de forme 3D générées.

Agrandir notre base de jeu de données (dataset) avec d’autres types de formes 3D qui possèdent plus de caractéristiques.

Utiliser d’autres architectures neuronales génératives (VAE) ou à base des transformateurs.

On conclure notre thème par des recommandations.

Ce system a été créé et dédié spécialement pour faciliter la construction des formes 3D au concepteur, en plus la génération des formes 3D à partir des descriptions textuelles garanti la qualité et la quantité et de gagner beaucoup de temps au niveau de la construction.

Cette application a été créée spécialement pour résoudre ce genre de problème.

Annexes

Les Annexes

Les informations en plus (Dictionnaires)

❖ N°1. La définition de CAD

CAD (Computer_aided Design) **[82]**, est l'utilisation de logiciels informatiques pour aider à la création, la manipulation, l'analyse ou l'optimisation d'une conception. Est utilisé pour générer des modèles virtuels 2Ds ou 3D.Les conceptions CAD 2D sont généralement destinées à l'ingénierie / l'architecture technique, tandis que les modèles 3D sont généralement utilisés pour des applications numériques tels que l'animation ou pour des processus de fabrication / prototypage comme l'impression 3D.

❖ N°2. La définition de Maillage 3D

Un maillage est une modélisation géométrique d'un domaine son principe et permet de relier un ensemble des éléments proportionnés fini, tel que son objectif est permis de procéder une simplification d'un system par un modèle qui représente ce dernier .

Le maillage est caractérisé par:

Le maillage est caractérisé par :

- son repère .
- ses points avec leur coordonnée .
- ses cellules qui constituent des polytopes reliant n de ces points.

et peut-être caractérisé notamment par :

- sa dimension : typiquement 2D ou 3D ;
- son volume (dimension totale couverte) ;
- sa finesse : surface ou volume moyen des cellules composant le maillage .
- la géométrie des cellules : triangles, quadrilatères (parallélogrammes, rectangles, carrés), …, polygones, en 2D ; tétraèdres, prismes, hexaèdres (parallélépipèdes, cubes), …, polyèdres en 3D .
- le degré de l'élément : c'est le degré du polynôme servant à décrire les côtés ou arêtes des éléments, un élément de degré 1 a des côtés ou arêtes rectilignes, dans le cas des éléments finis, c'est également le degré des polynômes d'interpolation.

❖ N°3. La définition de Amazon Mechanical Turk

Est un service de micro-travail lancer par Amzon.com **[83]** a la fin d'année de 2005, est une plateforme web de crowdsourcing son principe et permet d'effectuer par des humains, des tâches plus ou moins complexes, tel que les taches sont souvent d'analyse ou de produire de l'information dans le domaine de l'intelligence artificielle et l'analyse du contenu d'images.

❖ N°4. La définition de La tokenisation

Son principe est **[84]**consiste à découper ou bien transformer un texte en token, dans l'idée de chaque token représente un mot . un exemple sur la tokenisation de texte :

le texte :modern dining table chair with nice and sturdy for wooden legs. looks comfortable

- le token : ['modern', 'dining', 'table', 'chair', 'with' ,'nice', 'and', 'sturdy', 'for', 'wooden', 'legs' ,looks', 'comfortable'] .

❖ N°5. La définition de la lemmatisation

Est une méthode désignée **[85]** pour le traitement lexical de texte en vue de son analysé et pour le but de regrouper les mots d'une même famille .avec la forme canonique donnée pour les mots tels que chacun d'eux se trouve en une entité appelée en lexicologie lemme .

Les lemmes d'une langue utilisent plusieurs formes en fonction :

- du genre (masculin ou féminin).
- de leur nombre (un ou plusieurs).
- de leur personne (moi, toi, eux…).
- de leur mode (indicatif, impératif…).

❖ N°6. Pipelines spacy

Spacy (est une bibliothèque logicielle Python de traitement automatique des langues) offre une fonction NlP cette dernière et un point d'entrée vers toutes les fonctionnalités de Spacy.

Le pipeline **[86]** est utilisé dans le cas de traitement linguistique , par exemple si vous appelez le "NlP" sur un texte, tout d'abord spacy Toknisè le texte pour produire un objet Doc. Puis le document traité en plusieurs étapes cela signifier le pipeline de traitement tel que cette dernière est composée par : un tagué et un analyseur et d'un identificateur d'entité, on comprend à partir de cette partie chaque composant de pipeline renvoie le document traité qui est ensuite transmis au composant suivant.

❖ N°7. La définition de LanguageTool2

Est une combinaison d'Hunspell, et est un logiciel correcteur **[87]** des fautes d'orthographe et grammaire gratuit et open source a été lancé par Daniel Naber en 2003 écrit en python,» il est utilisé pour corriger les fautes de l'ensemble de description textuelle en langage naturel.

❖ N°8. Hunspell

Hunspell est un logiciel correcteur orthographique **[88]** open source basé sur Myspell, et en plus il est compatible avec les dictionnaires Myspell, il a ajouté une amélioration telle qu'elle l'encodage des dictionnaires en UTF-8 (Unicode) à la place de l'encodage en ASCII utilisé par Myspell.

❖ N°9. La définition de UNK token

Le type de chaine du jeton **[89]** n'est pas pratique à utiliser par le modèle qui possède une entrée numérique c'est pour cela ils ont construit un dictionnaire souvent appelé vocabulaire spécialement pour mapper les jetons (token) en indices numériques à partir de 0, il faut tout d'abord compter les jetons uniques dans tous les documents après attribuer pour chaque jeton unique un index numérique de fréquence tel que les jetons rarement apparus il faut les supprimer pour réduire la complexité .dans le cas où il existe des jetons supprimer ou bien qui n'existe pas dans le corpus sont transformés ou bien mappés sur un jeton spécial inconnu («Unk»).

- pour le remplissage on utilise le jeton «pad>»
- pour le début d'une phase on utilise le jeton «bos>»
- pour la fin d'une phase on utilise le jeton «Éos»

❖ N°10. La définition de la foulée (Stride)

Stride ou bien foulée **[90]** c'est le nombre de pixels décalés sur la matrice d'entrée .par exemple si la foulée et 1 nous déplaçons les filtres à 1 pixel à la fois et ainsi de suite on déplace les filtres selon le nombre de foulé.

Bibliographie

Bibliographie

[1] «Qu'est-ce que l'apprentissage automatique ? », IONOS Digitalguide. [En ligne]. Disponible sur: https://www.ionos.fr/digitalguide/web-marketing/analyse-web/quest-ce-que-lapprentissage-automatique/. ***[Consulté le: 09-mars-2020].***

[2] «Machine Learning: Tom M. Mitchell: 9780070428072: Amazon.com: Books ». [En ligne]. Disponible sur: https://www.amazon.com/Machine-Learning-Tom-M-Mitchell/dp/0070428077. ***[Consulté le: 09-mars-2020].***

[3] «L.Györfi, M. Kohler, A. Krzyzak, et H. Walk, *A Distribution-Free Theory of Nonparametric Regression*. » New York: Springer-Verlag, 2002.

[4] «Présentation ». [En ligne]. Disponible sur: http://www.college-de-france.fr/site/bibliotheques-archives/index.htm. ***[Consulté le: 09-mars-2020].***

[5] «L.Györfi, M. Kohler, A. Krzyzak, et H. Walk, *A Distribution-Free Theory of Nonparametric Regression* ». New York: Springer-Verlag, 2002.

[6] «Day 2 — Supervised Learning and Linear Regression - 30 days of Machine Learning - Medium ». [En ligne]. Disponible sur: https://medium.com/30-days-of-machine-learning/day-2-supervised-learning-and-linear-regression-3ed7b4abf0ad. ***[Consulté le: 06-mars-2020].***

[7] «Boughaba M et Boukhris B , L`apprentissage profond (Deep Learning) pour la classification et la recherche d'images par le contenu, Mémoire Master Professionnel, UNIVERSITE KASDI MERBAH OUARGLA,2017 ».

[8] «Machine Learning Explained: Understanding Supervised, Unsupervised, and Reinforcement Learning - Data Science Central | M.i.a., L intelligence ». [En ligne]. Disponible sur: https://www.pinterest.com/pin/356839970469739275/. ***[Consulté le: 06-mars-2020].***

[9] «Olivier Set Frédérick G, Comportements adaptatifs pour les agents dans des environnements informatiques complexes, Chapitre 1 Apprentissage par renforcement, Thèse d'Habilitation à diriger des recherches de l'université Paris VI,2004 »

[10] «O.SIGAUD, "Comportements adaptatifs pour les agents dans des environnements informatiques complexes," paris, 2004. »

[11] «R.Warlop»« Apprentissage par renforcement - Petit guide du machine learning », the tea house by fifty-five, 01-avr-2019. [En ligne]. Disponible sur: https://teahouse.fifty-five.com/fr/petit-guide-du-machine-learning-partie-4-lapprentissage-par-renforcement/. ***[Consulté le: 19-mars-2020].***

Bibliographie

[12] «Apprentissage par renforcement », *ResearchGate*. [En ligne]. Disponible sur: https://www.researchgate.net/figure/Apprentissage-par-renforcement_fig2_325929737. ***[Consulté le: 07-mars-2020].***

[13] «Bastien L, » « Réseau de neurones artificiels : qu'est-ce que c'est et à quoi ça sert ? », LeBigData.fr, 05-avr-2019. [En ligne]. Disponible sur: https://www.lebigdata.fr/reseau-de-neurones-artificiels-definition. ***[Consulté le: 06-mars-2020].***

[14] «Les neurones biologiques et artificiels | L'intelligence Artificielle – TPE ». [En ligne]. Disponible sur: https://iatpe2015.wordpress.com/le-fonctionnement/le-reseau-de-neurones-artificiel/les-neurones-biologiques-et-artificiels/. ***[Consulté le: 06-mars-2020].***

[15] «Les réseaux de neurones ». [En ligne]. Disponible sur: https://ml4a.github.io/ml4a/fr/neural_networks/. ***[Consulté le: 06-mars-2020].***

[16] «S.Polamuri, « Difference Between Softmax Function and Sigmoid Function », *Dataaspirant*, 07-mars-2017. [En ligne]. Disponible sur: https://dataaspirant.com/2017/03/07/difference-between-softmax-function-and-sigmoid-function/. ***[Consulté le: 06-mars-2020].***

[17] «SoftMax ». [En ligne]. Disponible sur: https://hannekedenouden.ruhosting.nl/RLtutorial/html/SoftMax.html. ***[Consulté le: 07-mars-2020].***

[18] «Le réseau de neurones artificiel - ppt video online télécharger ». [En ligne]. Disponible sur: https://slideplayer.fr/slide/5489483/. ***[Consulté le: 09-mars-2020].***

[19] «Classification-des-images-avec-les-reseaux-de-neurones.pdf ». .

[20] «Classification des images avec les réseaux de neurones convolutionnels - PDF Free Download ». [En ligne]. Disponible sur: https://docplayer.fr/79962650-Classification-des-images-avec-les-reseaux-de-neurones-convolutionnels.html. ***[Consulté le: 09-mars-2020].***

[21] « Les réseaux de neurones à convolution (CNN) ». https://www.bpesquet.fr/fr/slides/ai/convolutional-neural-networks/ ***[Consulté le sept. 16, 2020].***

[22] B.Shivam,«3D Convolutions : Understanding + Use Case | Kaggle ». https://www.kaggle.com/shivamb/3d-convolutions-understanding-use-case ***[Consulté le avr. 25, 2020].***

[23] « Process of 3D convolution layer. (a) 3D convolution of a feature... », *ResearchGate*. https://www.researchgate.net/figure/Process-of-3D-convolution-layer-a-3D-convolution-of-a-feature-map-with-a-filter-b_fig4_320260344 ***[Consulté le avr. 29, 2020]***

[24] «Analyse fine 2D/3D de véhicules par réseaux de neurones profonds ». https://www.researchgate.net/publication/323216752_Analyse_fine_2D3D_de_vehicules_par_reseaux_de_neurones_profonds *[Consulté l'avr. 09, 2020].*

[25] B.L.E.Habib,« Les réseaux de neurones convolutifs ». https://datasciencetoday.net/index.php/fr/deep-learning/173-les-reseaux-de-neurones-convolutifs (consulté le sept. 13, 2020).

[26] «Comment les Réseaux de neurones à convolution fonctionnent ». [En ligne]. Disponible sur: https://medium.com/@CharlesCrouspeyre/comment-les-r%C3%A9seaux-de-neurones-%C3%A0-convolution-fonctionnent-b288519dbcf8. *[Consulté le: 07-mars-2020].*

[27] «Découvrez les différentes couches d'un CNN - Classez et segmentez des données visuelles - OpenClassrooms ». [En ligne]. Disponible sur: https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun-cnn. *[Consulté le: 06-mars-2020].*

[28] C. Florian,« TEL - Thèses en ligne - Analyse fine 2D/3D de véhicules par réseaux de neurones profonds ». https://tel.archives-ouvertes.fr/tel-01708264 *[Consulté l'avr. 25, 2020].*

[29] S. Ioffe et C. Szegedy,« Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift », *arXiv:1502.03167 [cs]*, mars 2015, Consulté le: avr. 25, 2020. [En ligne]. Disponible sur: http://arxiv.org/abs/1502.03167.

[30] «Introducing: Flickr PARK or BIRD | code.flickr.com ». [En ligne]. Disponible sur: https://code.flickr.net/2014/10/20/introducing-flickr-park-or-bird/. *[Consulté le: 09-mars-2020].*

[31] «Que signifie Réseaux de neurones récurrents? - Definition IT de Whatis.fr ». [En ligne]. Disponible sur: https://www.lemagit.fr/definition/Reseaux-de-neurones-recurrents. *[Consulté le: 06-mars-2020].*

[32] «Recurrent Neural Net – nerdcoder ». https://nerdthecoder.wordpress.com/2019/02/03/recurrent-neural-net/ *[Consulté le sept. 16, 2020].*

[33] J.Nowak, A. Taspinar, et R. Scherer, « LSTM Recurrent Neural Networks for Short Text and Sentiment Classification », in *Artificial Intelligence and Soft Computing*, vol. 10246, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, et J. M. Zurada, Éd. Cham: Springer International Publishing, 2017, p. 553-562.

[34] Recurrent Neural Networks (RNN) | RNN LSTM | Deep Learning Tutorial | Tensorflow Tutorial Edureka

[35] H.Lasselin et G. Lecorvé, « Make text look like speech: disfluency generation using sequence-to-sequence neural networks », p. 18.

[36] «Implementation of RNN, LSTM, and GRU - Towards Data Science ». [En ligne]. Disponible sur: https://towardsdatascience.com/implementation-of-rnn-lstm-and-gru-a4250bf6c090. ***[Consulté le: 06-mars-2020].***

[37] «The internal structure of a gate recurrent unit (GRU). », *ResearchGate*. [En ligne]. Disponible sur: https://www.researchgate.net/figure/The-internal-structure-of-a-gate-recurrent-unit-GRU_fig2_301648210. ***[Consulté le: 07-mars-2020].***

[38] «Réseau antagoniste génératif », *Data Analytics Post*. [En ligne]. Disponible sur: https://dataanalyticspost.com/Lexique/reseau-antagoniste-generatif/.***[Consulté le: 09-mars-2020].***

[39] I.J. Goodfellow *et al.*, « Generative Adversarial Networks », *arXiv:1406.2661 [cs, stat]*, juin 2014.

[40] «[1611.02163] Unrolled Generative Adversarial Networks ». Consulté le: mai 30, 2020. [En ligne]. Disponible sur: https://arxiv.org/abs/1611.02163.

[41] «Common Problems | Generative Adversarial Networks », *Google Developers*. https://developers.google.com/machine-learning/gan/problems?hl=fr ***[Consulté le mai 30, 2020] .***

[42] K.Fukamizu, M. Kondo, et R. Sakamoto, « Generation High resolution 3D model from natural language by Generative Adversarial Network », *arXiv:1901.07165 [cs, stat]*, janv. 2019.

[43] chm,« A tutorial on Conditional Generative Adversarial Nets + Keras implementation », *mc.ai*, 17-juin-2019. [En ligne]. Disponible sur: https://mc.ai/a-tutorial-on-conditional-generative-adversarial-nets-keras-implementation/.***[Consulté le: 09-mars-2020].***

[44] J.Brownlee, « How to Develop a Wasserstein Generative Adversarial Network (WGAN) From Scratch », *Machine Learning Mastery*, 16-juill-2019. [En ligne]. Disponible sur: https://machinelearningmastery.com/how-to-code-a-wasserstein-generative-adversarial-network-wgan-from-scratch/. ***[Consulté le: 09-mars-2020].***

[45] «Information Theory and Statistics (Dover Books on Mathematics): Solomon Kullback: 9780486696843: Amazon.com: Books ». [En ligne]. Disponible sur: https://www.amazon.com/Information-Theory-Statistics-Dover-Mathematics/dp/0486696847. ***[Consulté le: 09-mars-2020].***

[46] «Performance Testing - The Complete Guide - Neotys ». [En ligne]. Disponible sur: https://www.neotys.com/insights/performance-testing. ***[Consulté le: 09-mars-2020].***

[47] P. C. Sanderson M., « Evaluating the performance of information retrieval systems using test collections », 15-juin-2013. [En ligne]. Disponible sur: http://informationr.net/ir/18-2/paper582.html#cro09. ***[Consulté le: 10-mars-2020].***

Bibliographie

[48] «Apprendre le Machine Learning en une semaine », *Machine Learnia*, 14-juin-2019. [En ligne]. Disponible sur: https://machinelearnia.com/apprendre-le-machine-learning-en-une-semaine/. ***[Consulté le: 09-mars-2020].***

[49] +Bastien L, « Confusion Matrix : l'outil de mesure de performances du Machine Learning », *LeBigData.fr*, déc. 10, 2018. https://www.lebigdata.fr/confusion-matrix-definition ***[Consulté l'août 09, 2020].***

[50] «Evaluation Metrics for Machine Learning - Accuracy, Precision, Recall, and F1 Defined », *Pathmind*. http://pathmind.com/wiki/accuracy-precision-recall-f1 ***[Consulté le août 09, 2020].***

[51] S. Reed, Z. Akata, H. Lee, et B. Schiele, « Learning Deep Representations of Fine-Grained Visual Descriptions », in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, p. 49-58, doi: 10.1109/CVPR.2016.13.

[52] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, et H. Lee, « Generative Adversarial Text to Image Synthesis », *arXiv:1605.05396 [cs]*, juin 2016.

[53] I. J. Goodfellow *et al.*, « Generative Adversarial Networks », *arXiv:1406.2661 [cs, stat]*, juin 2014.

[54] H. Zhang *et al.*, « StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks », *arXiv:1612.03242 [cs, stat]*, août 2017.

[55] T. Xu *et al.*, « AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks », in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, p. 1316-1324, doi: 10.1109/CVPR.2018.00143.

[56] C. B. Choy, D. Xu, J. Gwak, K. Chen, et S. Savarese, « 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction », *arXiv:1604.00449 [cs]*, avr. 2016.

[57] J. Wu, C. Zhang, T. Xue, W. T. Freeman, et J. B. Tenenbaum, « Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling », *arXiv:1610.07584 [cs]*, janv. 2017, Consulté le: août 25, 2020. [En ligne]. Disponible sur: http://arxiv.org/abs/1610.07584.

[58] K. Chen, C. B. Choy, M. Savva, A. X. Chang, T. Funkhouser, et S. Savarese, « Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings », *arXiv:1803.08495 [cs]*, mars 2018.

[59] «Deep NLP: Word Vectors with Word2Vec - Deep Learning Demystified - Medium ». https://medium.com/deep-learning-demystified/deep-nlp-word-vectors-with-word2vec-d62cb29b40b3 ***[Consulté le avr. 02, 2020].***

[60] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. (2013) 3111–3119.

[61] «A.Troynikov Shape Priors Part 2: Embedding Shapes - Discipline Beats Motivation ». http://troynikov.io/shape-priors-pca/ ***[Consulté le avr. 25, 2020].***

[62] P. Häusser, A. Mordvintsev, et D. Cremers, « Learning by Association - A versatile semi-supervised training method for neural networks », *arXiv:1706.00909 [cs]*, juin 2017, Consulté le: avr. 25, 2020. [En ligne]. Disponible sur: http://arxiv.org/abs/1706.00909.

[63] Kaya et Bilge, « Deep Metric Learning: A Survey », *Symmetry*, vol. 11, nº 9, p. 1066, août 2019, doi: 10.3390/sym11091066.

[64] J. Liu, F. Yu, et T. Funkhouser, « Interactive 3D Modeling with a Generative Adversarial Network », *arXiv:1706.05170 [cs]*, janv. 2018, Consulté le: août 25, 2020. [En ligne]. Disponible sur: http://arxiv.org/abs/1706.05170.

[65] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, et L. Guibas, « GRASS: Generative Recursive Autoencoders for Shape Structures », *ACM Trans. Graph.*, vol. 36, nº 4, p. 1-12, juill. 2017, doi: 10.1145/3072959.3073613.

[66] M. Mirza et S. Osindero, « Conditional Generative Adversarial Nets », *arXiv:1411.1784 [cs, stat]*, nov. 2014, Consulté le: août 25, 2020. [En ligne]. Disponible sur: http://arxiv.org/abs/1411.1784.

[67] M. Arjovsky, S. Chintala, et L. Bottou, « Wasserstein GAN », *arXiv:1701.07875 [cs, stat]*, déc. 2017, Consulté le: août 26, 2020. [En ligne]. Disponible sur: http://arxiv.org/abs/1701.07875.

[68] «Google Colab : Le guide Ultime | Le Data Scientist ». https://ledatascientist.com/google-colab-le-guide-ultime/ ***[Consulté le juill. 05, 2020].***

[69] «Python : définition et utilisation de ce langage informatique ». https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445304-python-definition-et-utilisation-de-ce-langage-informatique/ ***[consulté le juill. 05, 2020].***

[70] «TensorFlow : tout savoir sur la bibliothèque Machine Learning open source ». https://www.lebigdata.fr/tensorflow-definition-tout-savoir [***Consulté le juill. 05, 2020***]

[71] «Keras », *DeepAI*, mai 17, 2019. https://deepai.org/machine-learning-glossary-and-terms/keras ***[Consulté le juill. 05, 2020].***

[72] «Teem: nrrd: Definition of NRRD File Format ». http://teem.sourceforge.net/nrrd/format.html *[Consulté le juill. 05, 2020].*

[73] M. Everts, *pynrrd: Pure python module for reading and writing NRRD files.* .

[74] «What is NumPy? - Definition from Techopedia », *Techopedia.com*. https://www.techopedia.com/definition/33858/numpy *[Consulté le juill. 05, 2020].*

[75] M. Leplatre, *easydict: Access dict values as attributes (works recursively).*

[76] «pyymal - Recherche Google ». https://www.google.com/search?q=pyymal&oq=pyymal&aqs=chrome..69i57j0l7.4188j0j7&sourceid=chrome&ie=UTF-8 *[Consulté le juill. 05, 2020].*

[77] A.X. Chang *et al.*, « ShapeNet: An Information-Rich 3D Model Repository », *arXiv:1512.03012 [cs]*, déc. 2015, Consulté le: avr. 25, 2020. [En ligne]. Disponible sur: http://arxiv.org/abs/1512.03012.

[78] G. A. Miller, R. Beckwith, C. D. Fellbaum, D. Gross, K. Miller. 1990. WordNet: An online lexical database. Int. J. Lexicograph. 3, 4, pp. 235-244.

[79] «Overcome the simulation gap in 3D Deep Learning with Domain Randomization ». https://medium.com/vitalify-asia/sim2real-3d-domain-randomize-982f8f18a7d7 *[Consulté le avr. 29, 2020].*

[80] A.Karabassi, G. Papaioannou and T. Theoharis, "A Fast Depth-Buffer-Based Voxelization Algorithm," *Journal of Graphics Tools,* vol. 4, pp. 5-10, 2019.

[81] E.h. Soulaiman, S. Abderrahim and S. Khalid, "Multi-view passive 3D reconstruction: Comparison and evaluation of three techniques and a new method for 3D object reconstruction," *International Conference on Next Generation Networks and Services, NGNS*, pp. 194-201, 16 December 2014.

[82] M. Dahnert, A. Dai, L. J. Guibas, et M. Niessner, « Joint Embedding of 3D Scan and CAD Objects », présenté à Proceedings of the IEEE International Conference on Computer Vision, 2019, p. 8749-8758, Consulté le: avr. 25, 2020. [En ligne]. http://openaccess.thecvf.com/content_ICCV_2019/html/Dahnert_Joint_Embedding_of_3D_Scan_and_CAD_Objects_ICCV_2019_paper.html.

[83] «Amazon Mechanical Turk ». https://www.mturk.com/worker/help *[Consulté le mai 05, 2020].*

[84] M.Fabien, « Traitement Automatique du Langage Naturel en français (TAL / NLP) », *Stat4decision*, nov. 03, 2019. https://www.stat4decision.com/fr/traitement-langage-naturel-francais-tal-nlp/

[85] «Lemmatisation → Définition, c'est quoi lemmatiser ? | Facem Web ». https://facemweb.com/referencement-naturel-seo/lemmatisation *[Consulté le mai 05, 2020].*

[86] «Language Processing Pipelines · spaCy Usage Documentation », Language Processing Pipelines. https://spacy.io/usage/proc .

[87] «Language Tool - Spell and Grammar Checker », *LanguageTool*. https://languagetool.org/ ***[Consulté le mai 05, 2020].***

[88] «Hunspell: About ». http://hunspell.github.io/ ***[Consulté le mai 06, 2020].***

[89] «8.2. Text Preprocessing — Dive into Deep Learning 0.7.1 documentation ». https://d2l.ai/chapter_recurrent-neural-networks/text-preprocessing.html ***[Consulté le mai 06, 2020].***

[90] Prabhu, « Understanding of Convolutional Neural Network (CNN) — Deep Learning », *Medium*, nov. 21, 2019. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 ***[Consulté le mai 18, 2020].***