REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB BLIDA

FACULTE DES SCIENCES

DEPARTEMENT D'INFORMATIQUE

Mémoire De Fin D'études

Pour l'obtention du Diplôme de Master en Informatique

Option : Système informatique et réseaux

THÈME

Génération de forme 3D en utilisant une architecture encoder-encoder

Présenté Par :

Otmane Telba Amina & Seghiri Ihcene

Soutenu Publiquement le : 01/07/2024

Devant le jury composé de :

Encadreur: Mr A.H KAMECHE

Président (e) : Mme H.Ykhlef

Examinateur : Mme Kasri

Année Universitaire : 2023/2024

Résumé

De nos jours, la modélisation 3D est devenu l'un des domaines les plus régnant, allant des applications médicales essentielles aux domaines de divertissement comme les jeux vidéo et les effets visuels pour le cinéma. Cependant, la création manuelle de modèles 3D complexes est une tâche fastidieuse et chronophage, nécessitant des compétences et une expertise considérable en matière de logiciels de modélisation 3D.

L'objectif de ce projet est de développer une approche innovante permettant de générer automatiquement des modèles 3D à partir d'une simple description textuelle. Cette méthode vise à faciliter et à accélérer le processus de modélisation 3D, en offrant aux utilisateurs la possibilité de créer des formes 3D complexes en fournissant simplement une description textuelle de l'objet souhaité.

Grâce à des techniques d'intelligence artificielle, de traitement automatique du langage naturel (NLP) et d'apprentissage automatique, notre système analyse les descriptions textuelles fournies et génère des modèles 3D correspondants. Pour réaliser cela, nous devons passer par deux méthodes principales. La première utilise un réseau générateur conditionnel (CGAN) avec du bruit pour générer les modèles à partir de deux datasets : ShapeNet, basé sur les meubles et les chaises, et Primitives, contenant des formes simples telles que les cubes. La seconde méthode repose sur des modèles de diffusion appliqués à ces mêmes datasets.

Mots clé: Modélisation 3D, description textuelle, intelligence artificielle, traitement automatique du langage naturel(NLP), CGAN.ShapeNet.

Abstract:

Nowadays , 3D modeling has become one of the most important fields, ranging from essential medical applications to entertainment domains such as video games and visual effects for cinema. However, manually creating complex 3D models is a tedious and time-consuming task, requiring considerable skills and expertise in 3D modeling software.

The goal of this project is to develop an innovative approach to automatically generate 3D models from a simple textual description. This method aims to facilitate and accelerate the 3D modeling process by allowing users to create complex 3D shapes by simply providing a

textual description of the desired object.

Using artificial intelligence techniques, natural language processing (NLP), and machine learning, our system analyzes the provided textual descriptions and generates corresponding 3D models. To achieve this, we must go through two main methods. The first uses a Conditional Generative Adversarial Network (CGAN) with noise to generate the models from two datasets: ShapeNet, based on tables and chairs, and Primitives, containing simple shapes such as cubes. The second method relies on diffusion models applied to these same datasets.

***Keywords*:** 3D modeling, textual description, artificial intelligence, natural language processing (NLP), CGAN, ShapeNet.

ملخص :

بدءًا من التطبيقات الطبية الأساسية إلى مجالات الترفيه مثل ، أصبحت النمذجة ثلاثية الأبعاد واحدة من أكثر المجالات السائدة،اليوم ، يعد إنشاء نماذج ثلاثية الأبعاد معقدة يدويًا مهمة شاقة وتستغرق وقتًا طويلاً،ألعاب الفيديو والمؤثرات البصرية للسينما. ومع ذلك وتتطلب مهارة وخبرة كبيرة في برامج النمذجة ثلاثية الأبعاد.

الهدف من هذا المشروع هو تطوير نهج مبتكر لإنشاء نماذج ثلاثية الأبعاد تلقائيًا من وصف نصي بسيط. تهدف هذه الطريقة إلى جعل مما يوفر للمستخدمين القدرة على إنشاء أشكال ثلاثية الأبعاد معقدة بمجرد تقديم وصف ،عملية النمذجة ثلاثية الأبعاد أسهل وأسرع نصي للكائن المطلوب.

يقوم نظامنا بتحليل أوصاف النص المقدمة ،باستخدام الذكاء الاصطناعي ومعالجة اللغة الطبيعية وتقنيات التعلم الآلي (NLP) يجب أن نتبع طريقتين رئيسيتين. يستخدم الأول شبكة المولدات الشرطية،وإنشاء نماذج ثلاثية الأبعاد مقابلة. ولتحقيق ذلك (CGAN) مع الضوضاء لإنشاء نماذج من مجموعتي بيانات :ShapeNet،المستندة إلى الأثاث والكراسي ،Primitives و، التي تحتوي على أشكال بسيطة مثل المكعبات. وتعتمد الطريقة الثانية على نماذج الانتشار المطبقة على مجموعات البيانات نفسها.

الكلمات الرئيسية :

نمذجة ثلاثية الأبعاد، وصف نصي، ذكاء اصطناعي، معالجة اللغة الطبيعية، تعلم آلي

Remerciements

Nous souhaitons exprimer notre profonde gratitude envers notre Dieu miséricordieux, qui nous a donné l'opportunité, la santé, la force intérieure et la patience nécessaire pour réalisé ce travail.

Nous tenons à nous remercier infiniment pour toutes sortes d'efforts déployés pour surmonter tous les défis afin d'atteindre ce stade.

Nous souhaitons également exprimer notre gratitude envers toutes les personnes qui nous ont apporté leur soutien, de quelque manière que ce soit, pour nous permettre d'en arriver là.

Un immense remerciement à notre encadreur, Monsieur KAMECHE, qui nous a accompagné et guidé tout au long de cette expérience qui a été enrichissante grâce à lui.

Un grand merci à tous les enseignants du département informatique de Blida et à tous les employés de l'université Saad Dahleb qui ont tous contribué à notre formation intéressante.

Nous souhaitons également adresser nos remerciements aux membres du jury pour avoir minutieusement évalué notre travail.

Ihcene & Amina

Dédicace

Je dédie ce travail avec tout mon cœur à ma famille, qui m'a accompagnée sans faiblir ces dernières années, et qui a été mon soutien inébranlable dans les moments les plus difficiles. Ma mère, cette lumière qui éclaire mon chemin, vous êtes à l'origine de tout, votre amour et vos prières m'ont permis de trouver le courage de poursuivre mes rêves. Mon père, grand-mère, sœurs et frère, vous avez tous fait preuve de générosité et de patience, et je suis reconnaissante de votre soutien.

Je dédie ce travail à mon professeur et à ma tante, Zahira, qui m'a appris et m'a changé pour le mieux à certains moments de ma vie, et son influence et son amour restent dans mon cœur pour toujours, ses conseils et son soutien illimités depuis l'enfance.

Je tiens également à exprimer ma sincère gratitude envers mon binôme, Amina, pour son travail acharné, sa collaboration exceptionnelle et notre complicité tout au long de ce projet. Ensemble, nous avons surmonté les défis, partagé nos idées et atteint des résultats qui dépassent nos attentes. Ce succès est le fruit de notre collaboration solide et je suis fière de ce que nous avons accompli ensemble.

Je tiens à remercier tous mes amis, en particulier Ilham, Sara, Wissam et Tinhinane, ainsi que tous ceux qui m'ont apporté leur soutien et leurs encouragements au fil des années. Votre présence, vos encouragements et votre amitié ont rendu ce parcours plus agréable et significatif.

À tous ceux qui ont contribué à mon parcours académique, je vous suis profondément reconnaissante pour votre soutien, votre confiance et votre amour. Votre impact sur ma vie dépasse largement les mots et restera gravé dans ma mémoire pour toujours. Merci infiniment.

Avec amour et reconnaissance

Ihcene

Dédicace

Je dédie ce modeste travail, A mes précieux parents et mes très chères sœurs et frère,
Dont l'amour inconditionnel et le soutien indéfectible ont été mon ancre tout au long de ce cheminement. Votre présence réconfortante et vos encouragements perpétuels m'ont permis de persévérer.
A ma binôme, pour sa contribution remarquable à ce projet, Et pour avoir été un véritable soutien moral dans les moments de doute et de remise en question. Ta patience, ton écoute et ta motivation ont été des atouts inestimables.
A mes très chers amis Nesrine, Sheraz, Sara, Wissam et Tinhinane, Dont la présence chaleureuse, les rires partagés et les encouragements ont été une véritable bouffée d'oxygène. Votre amitié m'a donné la force d'avancer avec sérénité.
Je dédie ce travail à toutes les personnes qui ont cru en moi et en mes compétences,Et qui, de près ou de loin, ont participé à la réalisation de ce mémoire. Votre confiance en mes capacités a été un moteur essentiel.

Amina

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

Abbreviation	Description complete
CNN	Convolutional neural Network
RNN	Recurrent neural Network
GAN	Generative adversial Network
CGAN	Conditional Generative adversial Network
BERT	Bidirectional Encoder Representation from Transformers
SBERT	Sentence Bidirectional Encoder Representation from Transformers
LSTM	Long Short Term Memory
ReLU	Rectified Linea Unit
NLP	natural language processing
SNLI	Stanford Natural Language Inference
MNLI	Multi-Genre Natural Language Inference
MSELoss	Mean Squared Error Loss
NumPy	Numerical Python

Introduction générale

La modélisation 3D s'est imposée comme un domaine stratégique, avec des applications majeures dans les sciences médicales, les loisirs numériques et les effets spéciaux cinématographiques. Un défi supplémentaire réside dans la capacité à imaginer et concevoir des formes inédites, puis à les traduire en descriptions textuelles, en vue d'une génération automatique. C'est pourquoi d'intenses efforts de recherche visent sans cesse à optimiser et perfectionner ces techniques de pointe à l'interface entre le visuel et le langage naturel.[1]

Problématique

La création des formes 3D est difficile et complexe dans la méthode traditionnelle, nécessitant une expertise et une intervention manuelle importantes. Cependant, la génération de formes 3D à partir de descriptions textuelles offre une solution potentielle à ce problème, mais pose un défi supplémentaire quant à la capacité de générer des formes 3D à partir de descriptions textuelles qui ne font pas partie du jeu de données d'entraînement. La problématique consiste à développer un modèle capable de résoudre cette difficulté, en apprenant les relations entre le texte et les formes 3D pour générer des modèles précis et réalistes, tout en garantissant la flexibilité du modèle pour s'adapter à une grande variété de descriptions textuelles inconnues lors de l'entraînement, notamment pour des objets courants tels que les tables et les chaises.

Objectif

L'objectif principal est de mettre en œuvre un système d'intelligence artificielle capable de comprendre et d'interpréter les descriptions textuelles pour générer des modèles 3D précis, réalistes et flexibles, applicable à une grande variété de descriptions textuelles inconnues lors de l'entraînement.Pour atteindre cet objectif, plusieurs défis doivent être relevés, notamment la compréhension du langage naturel, la représentation 3D, la génération de modèles, la gestion des ambiguïtés et l'interaction utilisateur.

Ce projet doit proposer des solutions combinant des techniques de traitement du langage naturel, d'apprentissage automatique et de modélisation géométrique.

Notre mémoire sera divisé comme ceci :

Chapitre 1: Dans ce chapitre, nous introduirons les notions de base et les travaux connexes en les comparant.

Chapitre 2: Ce chapitre contiendra l'architecture de notre travail ainsi que les méthodes que nous avons employées pour le réaliser.

Chapitre 3 : Le dernier chapitre contiendra les outils et les jeux de données utilisés, ainsi que les expérimentations que nous avons menées et les résultats obtenus.

Chapitre 1:notion de base

1. Introduction

La génération de modèles 3D à partir de descriptions textuelles a connu un essor significatif ces dernières années. Cette approche vise à créer des systèmes capables de comprendre et d'interpréter des descriptions textuelles, puis de les transformer en représentations 3D correspondantes. Elle permet ainsi de faciliter et d'accélérer le processus de modélisation 3D, traditionnellement fastidieux et nécessitant des compétences pointues.

Ce chapitre a pour objectif de présenter les différentes notions et techniques liées à la génération de formes 3D à partir de descriptions textuelles. En particulier, nous aborderons les concepts d'apprentissage automatique et d'apprentissage profond qui sont au cœur de cette approche appelée "text-to-shape".

Partie 1 : notion de base en machine learning

2. l'apprentissage automatique (ou Machine learning en anglais)

Le machine learning (ML) est une forme d'intelligence artificielle (IA) qui est axée sur la création de systèmes qui apprennent, ou améliorent leurs performances, en fonction des données qu'ils traitent. L'intelligence artificielle est un terme large qui désigne des systèmes ou des machines simulant une forme d'intelligence humaine. Le machine learning et l'IA sont souvent abordés ensemble et ces termes sont parfois utilisés de manière interchangeable bien qu'ils ne renvoient pas exactement au même concept. Une distinction importante est que, même si l'intégralité du machine learning repose sur l'intelligence artificielle, cette dernière ne se limite pas au machine learning, Nous allons parler des trois approches de l'apprentissage de l'apprentissage automatique[2].

2.1. Machine learning supervisé

Les algorithmes de machine learning supervisé sont les plus couramment utilisés. conclusions qu'il doit tirer. Tout comme un enfant apprend à identifier les fruits en les mémorisant dans un imagier, en apprentissage supervisé, l'algorithme apprend grâce à un jeu de données déjà étiqueté et dont le résultat est prédéfini, La figure 1 montre un exemple d'apprentissage supervisé [3].

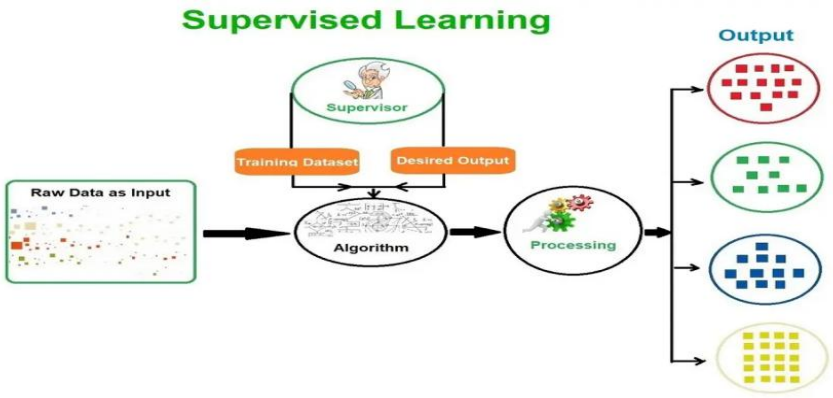


Figure 1- Machine learning supervisé [4]

2.2. Machine learning non supervisé

Cette branche de l'apprentissage automatique consiste à trouver des transformations intéressantes des données d'entrée sans l'aide d'aucune cible, dans le but de visualiser les données, de les compresser ou de les débruiter, ou pour mieux comprendre les corrélations présentes dans les données en question. L'apprentissage non supervisé est le pain quotidien de l'analytique des données, et c'est souvent une étape nécessaire pour mieux comprendre un ensemble de données avant d'essayer de résoudre un problème d'apprentissage supervisé. La réduction de dimension et le clustering sont des catégories bien connues d'apprentissage non supervisé, La figure 2 montre un exemple d'apprentissage non supervisé [5].

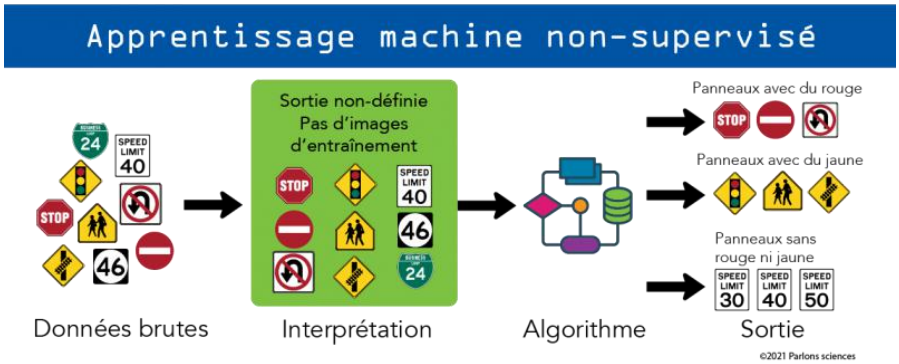


Figure 2-Machine learning non supervisé [6].

2.3. Apprentissage par renforcement (Reinforcement learning)

Cette branche de l'apprentissage automatique, longtemps négligée, Dans l'apprentissage par renforcement, un agent reçoit des informations sur son environnement et apprend à choisir les actions qui maximiseront sa récompense. Par exemple, un réseau de neurones analysant l'écran d'un jeu vidéo et produisant des actions de jeu pour maximiser le score peut être entraîné par renforcement.

Actuellement, l'apprentissage par renforcement reste principalement un domaine de recherche, sans grands succès pratiques en dehors des jeux vidéo. Mais on s'attend à ce qu'il prenne de l'importance dans de nombreuses applications réelles: voitures autonomes, robotique, gestion de ressources, éducation, etc. C'est une idée dont le temps est arrivé ou arrivera bientôt, La figure 3 montre un exemple d'Apprentissage par renforcement [7].

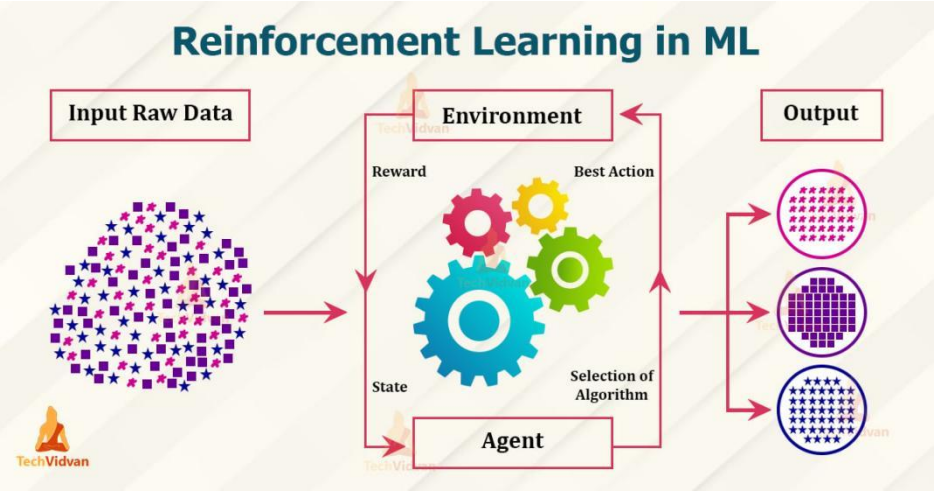


Figure 3- Apprentissage par Renforcement [8].

3. Apprentissage profond (deep learning)

L'apprentissage profond (deep learning) est un sous-domaine de l'apprentissage automatique qui repose sur des réseaux de neurones artificiels composés de multiples couches. Ces réseaux sont entraînés à extraire des représentations hiérarchiques de données complexes comme les images, le texte ou les données 3D.

Le processus fonctionne comme suit : les données brutes d'entrée (par exemple, une image) sont transmises à la première couche du réseau. Cette couche applique alors une transformation non linéaire pour extraire des caractéristiques de bas niveau (comme des contours ou des textures). L'output de cette première couche est ensuite transmis à la couche suivante, qui extrait des caractéristiques plus abstraites à partir des caractéristiques de bas niveau. Ce processus se répète à travers les couches successives du réseau, chaque couche construisant des représentations de plus en plus abstraites et complexes. Au final, les dernières couches du réseau produisent une représentation très abstraite des données d'entrée, qui peut être utilisée pour diverses tâches comme la classification, la détection d'objets ou la génération de nouvelles données, La figure 4 montre un exemple d'Apprentissage profond [9].

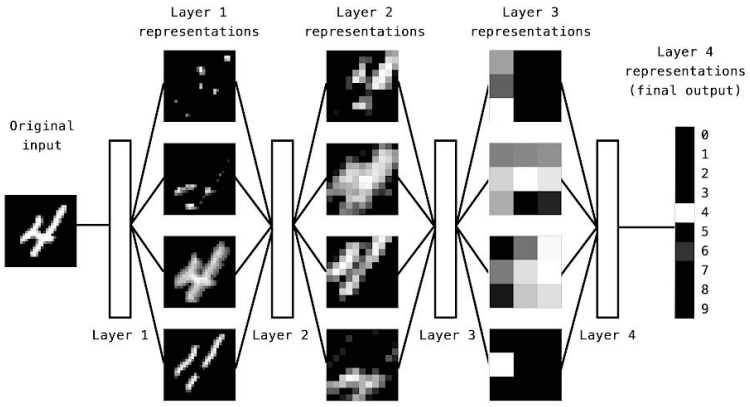


Figure 4-Un réseau neuronal profond pour la classification des chiffres [10].

3.1. Fonctions d'activation

Les fonctions d'activation sont des éléments essentiels dans les réseaux de neurones artificiels, car elles introduisent une non-linéarité qui permet au modèle d'apprendre et de modéliser des

relations complexes entre les données d'entrée et de sortie. On peut classer les fonctions d'activation en deux catégories principales : les fonctions linéaires et les fonctions non linéaires, La figure 5 montre les types de fonctions d'activation :

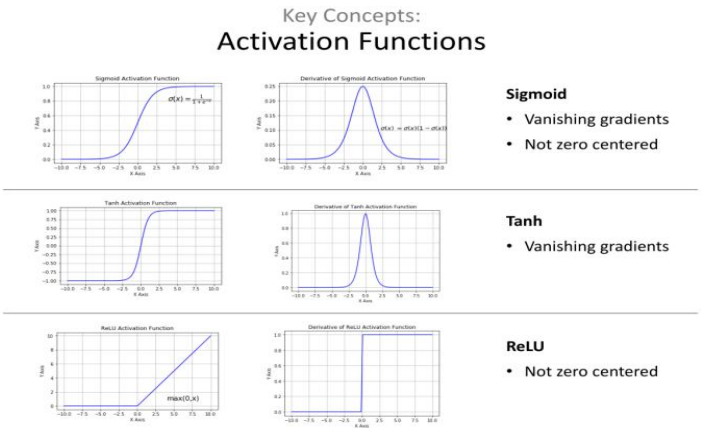


Figure 5- Types de fonctions d'activation [11].

3.1.1. Fonctions d'activation linéaires

Ces fonctions appliquent simplement une transformation linéaire aux données d'entrée. Elles ne sont généralement pas utilisées dans les réseaux de neurones profonds, car elles limitent la capacité du modèle à apprendre des relations non linéaires. Exemples : fonction identité (f(x) = x), fonction constante (f(x) = c), La figure 6 représente un diagramme de fonctions linéaires [12].

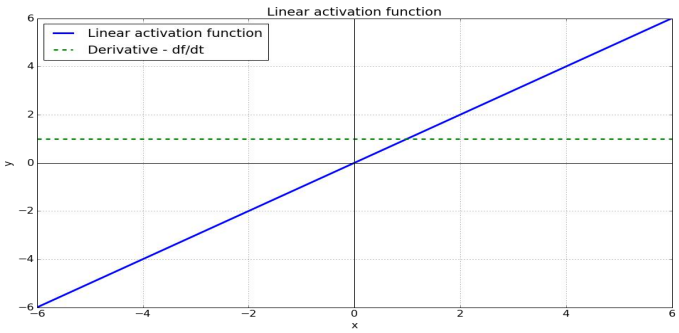


Figure 6- Fonctions linéaires [13].

3.1.2. Fonctions d'activation non linéaires

Ces fonctions introduisent une non-linéarité cruciale qui permet aux réseaux de neurones de modéliser des relations complexes , La figure 7 représente un diagramme de fonctions non linéaires [14].

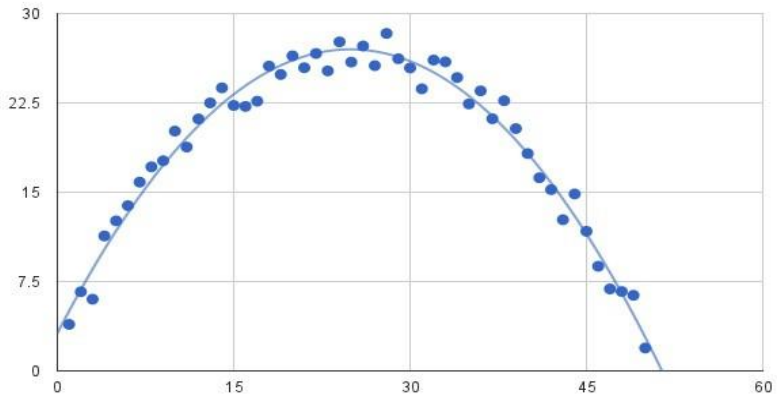


Figure 7- Fonctions non linéaires [15]

3.2. Fonction perte (Loss Function)

La fonction perte est une mesure qui quantifie l'erreur commise par le modèle sur les données d'entraînement. Elle compare les prédictions du modèle aux valeurs réelles ou attendues. L'objectif est de minimiser cette fonction perte pendant l'entraînement du réseau de neurones [16].

3.3. Descente de gradient (Gradient Descent)

La descente de gradient est un algorithme d'optimisation utilisé pour minimiser la fonction perte en mettant à jour les poids du réseau de neurones de manière itérative. À chaque itération, les gradients de la fonction perte par rapport aux poids sont calculés, et les poids sont ajustés dans la direction opposée au gradient, avec un taux d'apprentissage qui contrôle la taille des mises à jour [17].

4. L'architecture de réseau de neurones (neural network architecture)

fait référence à la structure et à l'agencement des différentes couches et connexions d'un réseau de neurones artificiels. Elle détermine la manière dont les données d'entrée sont transformées et propagées à travers le réseau pour produire les sorties souhaitées. Les principaux éléments clés de l'architecture comprennent le nombre de couches cachées, le nombre de neurones par couche, les fonctions d'activation utilisées, les types de connexions (feedforward, récurrentes, etc.), ainsi que la façon dont les couches sont interconnectées. Une architecture bien conçue est essentielle pour permettre au réseau d'apprendre efficacement à

partir des données et de généraliser correctement sur de nouvelles données, La figure 8 représente L'architecture de réseau de neurones [18].

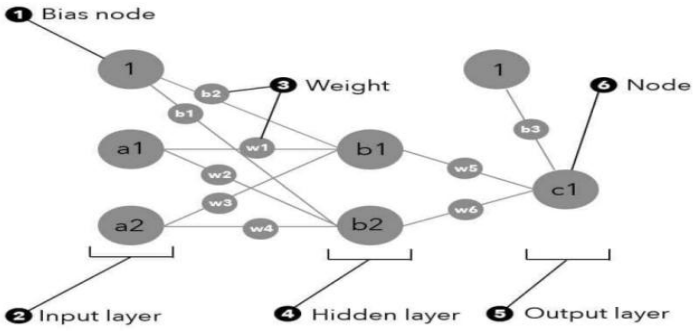


Figure 8- L'architecture de réseau de neurones [19].

Nous présentons dans la figure 9 un aperçu des différents types de réseaux de neurones artificiels abordés dans ce chapitre :

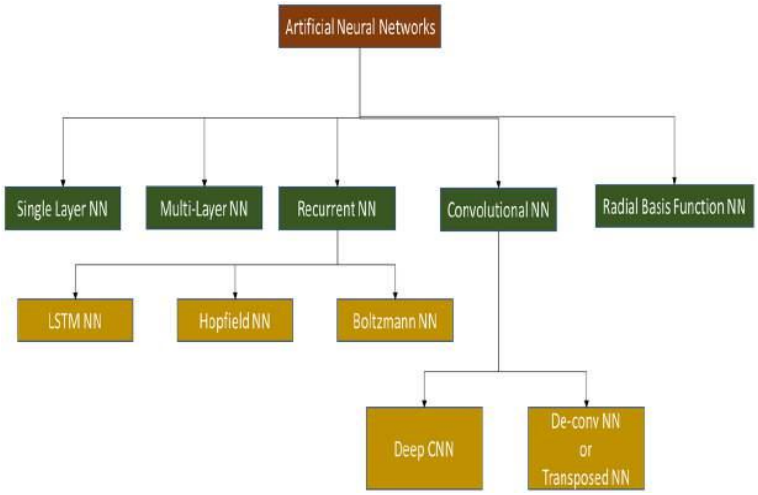


Figure 9 -Types de réseaux de neurones artificiel [20].

5. Les algorithmes de Apprentissage profond (deep learning)

Les algorithmes de deep learning, également connus sous le nom d'apprentissage profond, sont une branche de l'apprentissage automatique qui s'inspire du fonctionnement du cerveau humain pour traiter des données. Ils utilisent des réseaux de neurones artificiels composés de multiples couches interconnectées pour apprendre des représentations hiérarchiques de données brutes. Voici une explication des principaux algorithmes de deep learning

5.1. Réseaux de neurones convolutionnels (CNN)

Les réseaux de neurones convolutionnels (CNN) sont spécialement conçus pour reconnaître des motifs visuels directement à partir de données brutes comme des images, en appliquant des opérations qui préservent les relations spatiales. Leur principale utilisation se situe dans le domaine de l'imagerie, où ils permettent la classification, la détection et la reconnaissance d'objets présents dans des données d'images ou de vidéos, La figure 10 montre un exemple de réseaux de neurones convolutionnels [21].

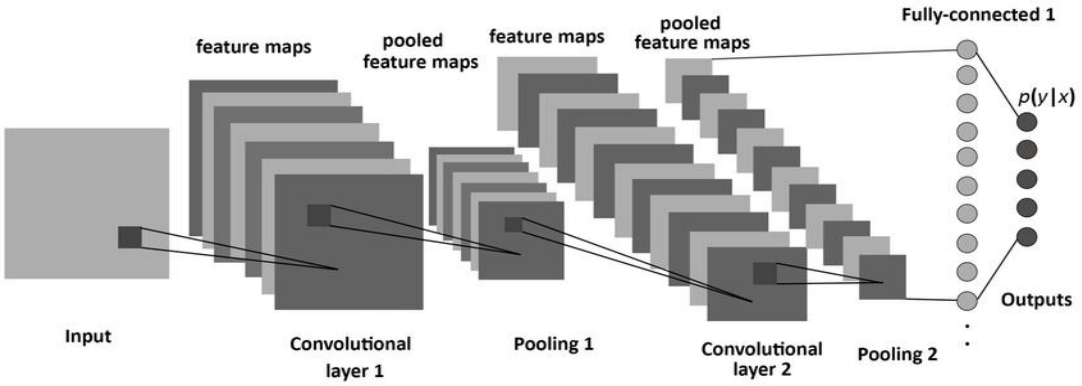


Figure 10- Exemple qui illustre le Réseaux de neurones convolutionnels [22].

5.2. Réseaux de neurones récurrents (RNN)

Un réseau de neurones récurrent (RNN) combine son entrée actuelle avec ses états de mémoire précédents, lui permettant ainsi de traiter arbitrairement des séquences d'entrées de longueur variable tout en conservant en mémoire un état interne représentant l'information pertinente extraite jusqu'alors dans la séquence. Les RNN ont trouvé de nombreuses applications, notamment dans le traitement du langage naturel, la reconnaissance de la parole, le traitement de séries temporelles, la traduction automatique, l'imagerie, etc. car ils sont particulièrement adaptés aux tâches impliquant des données séquentielles, grâce à leur capacité à intégrer l'information de leur état de mémoire précédent à leur entrée actuelle et à capturer ainsi l'information pertinente extraite au fil de la séquence au sein d'un état interne,
La figure 11 montre un exemple de réseaux de neurones récurrents [23].

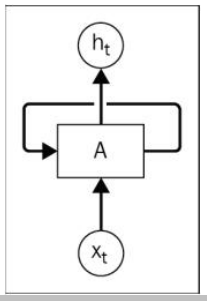


Figure 11- Exemple qui illustre un Réseau de neurones récurrent [24].

5.3. l'architecture des auto-encodeurs

Les auto-encodeurs sont un type de réseaux de neurones non supervisés constitués d'un encodeur et d'un décodeur. L'encodeur compresse les données d'entrée en une représentation de plus faible dimension appelée code ou représentation latente. Le décodeur reconstruit ensuite les données d'entrée à partir de cette représentation latente. L'architecture typique comprend une couche d'entrée, une ou plusieurs couches cachées pour l'encodeur qui réduisent progressivement la dimensionnalité, une couche de code de basse dimension, puis des couches cachées de décodage qui font l'opération inverse pour reconstruire la sortie. Le but est que le réseau apprenne une représentation compacte et utile des données d'entrée au niveau de la couche de code latente. Les variations comme les auto-encodeurs dénoisants, contractants ou variationnels permettent d'apprendre des représentations robustes ou structurées différemment [25].

5.3.1. L'encodeur

L'encodeur est la partie du réseau d'auto-encodeurs qui transforme les données d'entrée de grande dimension en une représentation de plus basse dimension appelée code ou représentation latente. Il est composé de couches successives, généralement des couches denses complètement connectées ou des couches convolutionnelles, qui appliquent des transformations non linéaires aux données pour en extraire les caractéristiques les plus pertinentes et compactes [26].

5.3.2. Le décodeur

est la partie complémentaire qui reconstruit une approximation des données d'entrée originales à partir du code latent produit par l'encodeur. Il est constitué de couches miroirs par

rapport à l'encodeur, qui appliquent des transformations inverses couche par couche pour aller de la représentation de basse dimension vers la dimension d'origine des données reconstruites [27].

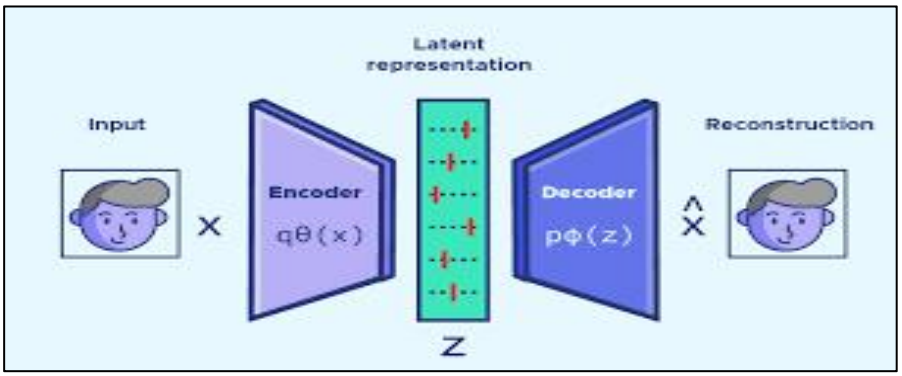


Figure 12-exemple qui illustre l'architecture des auto-encodeurs [28].

5.4. Réseaux antagonistes génératifs (GAN)

Un réseau antagoniste génératif (GAN) est une architecture de deep learning innovante. Il entraîne simultanément deux réseaux de neurones en les opposant, dans le but de générer de nouvelles données authentiques à partir d'un jeu de données d'entraînement donné. Par exemple, vous pouvez générer de nouvelles images réalistes à partir d'une base de données d'images existantes, ou de la musique originale à partir d'une base de données de chansons. Un GAN est qualifié d'antagoniste car il met en compétition deux réseaux différents.

Le générateur est un réseau qui génère de nouvelles données en échantillonnant des données d'entrée et en les modifiant de manière réaliste.

L'autre réseau, ***le discriminateur***, essaie de prédire si les données générées proviennent du jeu de données d'origine ou non. En d'autres termes, le discriminateur détermine si les données générées sont fausses ou réelles. Le système génère des versions améliorées des fausses données jusqu'à ce que le discriminateur ne puisse plus les distinguer des données originales [29], Un GAN est qualifié d'antagoniste car il met en compétition deux réseaux différents : le générateur, qui génère de nouvelles données en échantillonnant des données d'entrée et en les modifiant de manière réaliste, et le discriminateur, qui essaie de prédire si les données générées proviennent du jeu de données d'origine ou non. Le système génère des versions améliorées des fausses données jusqu'à ce que le discriminateur ne puisse plus les distinguer des données originales, ce qui peut être décrit mathématiquement par la formule

$$min_G max_D\, V(D,G)$$

$$V(D,G) = E_{x\sim pdata(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z))]$$

où le générateur essaie de minimiser la perte du discriminateur et le discriminateur essaie de minimiser sa propre perte

Où,

G = générateur ; D = discriminateur ; D(x) = sortie du réseau discriminateur pour l'échantillon x ; G(z) = sortie du réseau générateur pour l'échantillon z

Pdata(x) = distribution des données réelles

P(z) = distribution du générateur

x = échantillon de Pdata(x)

z = échantillon provenant de P(z)

Voici un exemple d'architecture de GAN, comme l'illustre la figure 13 :

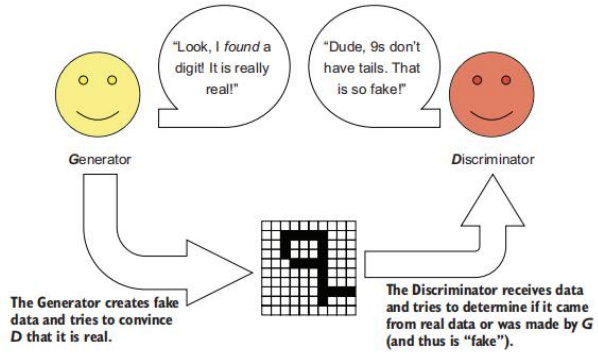


Figure 13 -Un exemple de haute qualité montre l'interaction entre le générateur et le discriminateur[30].

Un autre exemple d'architecture de CGAN est illustré par la figure 14 :

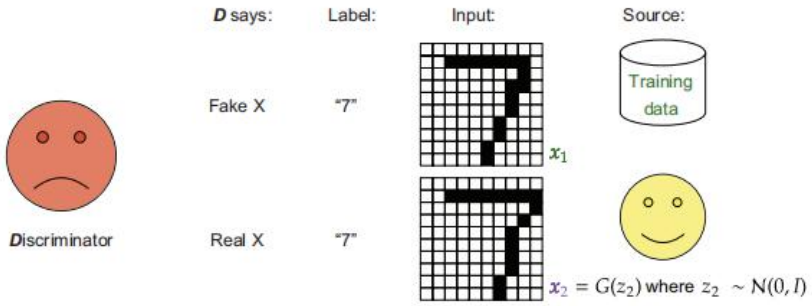


Figure 14- Exemple de GAN conditionnel où D a fait des erreurs et G a bien fait[31].

5.5. Les réseaux de mémoire à terme (LSTM)

Les réseaux de mémoire à long terme (LSTM) sont un type spécialisé de réseaux neuronaux

récurrents (RNN) qui répondent au défi de la dépendance à long terme dans les RNN traditionnels. Les LSTM sont conçus pour conserver les informations sur des séquences étendues, ce qui les rend idéaux pour les tâches où les informations passées influencent de manière significative les prévisions futures. Le composant central des LSTM est l'état de la cellule, qui facilite le flux d'informations de la mémoire du bloc précédent vers la mémoire du bloc actuel. En contrôlant la quantité d'informations précédentes à conserver via des couches spécifiques du module LSTM, ces réseaux peuvent capturer et utiliser efficacement les données historiques pour des prévisions précises. Grâce à leur architecture unique comprenant des fonctions d'activation de portes et des fonctions d'activation de sortie, les LSTM excellent dans l'apprentissage et la mémorisation d'informations sur de longues périodes, ce qui en fait un outil précieux pour les tâches nécessitant la mémoire d'événements passés dans les réseaux de neurones [32] voire la figure 15.

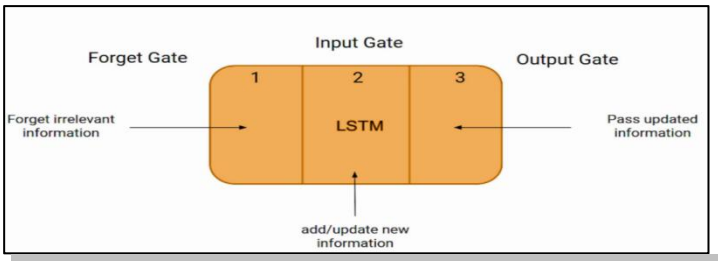


Figure 15 - Composants clés d'une cellule LSTM [33].

5.6. Modèles de diffusion

Un modèle de diffusion est une chaîne de Markov paramétrée formée à l'aide d'inférence variationnelle pour produire des échantillons correspondants aux données après un temps fini. Inspirés du principe de fonctionnement et des fondements mathématiques d'un modèle probabiliste, ces modèles analysent et prévoient le comportement d'un système qui varie dans le temps. En formant les modèles à l'aide de l'inférence variationnelle, des calculs complexes sont effectués pour trouver les paramètres exacts de la chaîne de Markov qui correspondent aux données observées. Une fois formé, le modèle peut générer des échantillons représentant des trajectoires possibles du système, avec chaque trajectoire ayant une probabilité différente de se produire. Les modèles de diffusion sont également des modèles génératifs profonds qui fonctionnent en ajoutant du bruit aux données d'apprentissage, puis en inversant le processus

pour récupérer les données, ce qui apprend progressivement à supprimer le bruit et génère ainsi de nouvelles images de haute qualité à partir de graines aléatoires.

Les modèles de diffusion sont divisés en trois catégories fondamentales : les modèles probabilistes de diffusion de débruitage (DDPM), les modèles génératifs basés sur les scores (SGM) conditionnés par le bruit et les équations différentielles stochastiques (SDE). Les DDPM sont principalement utilisés pour supprimer le bruit des données visuelles ou audio, tandis que les SGM peuvent générer de nouveaux échantillons à partir d'une distribution donnée. Les SDE décrivent les changements dans les processus aléatoires concernant le temps. Les modèles de diffusion ont des applications variées en intelligence artificielle, notamment la génération de vidéos de haute qualité, la création de vidéos haut de gamme, la génération de texte en image et la génération d'images photoréalistes. Les modèles de diffusion peuvent également être utilisés pour étendre les ensembles de données sur les soins de santé, pour générer des visages de célébrités de haute qualité et pour produire des images réalistes en ajoutant de manière transparente des images générées par l'IA entre les images réelles[35].

6. Mesures de performance et évaluation du modèle

L'évaluation des performances d'un modèle est une tâche à la fois critique et complexe. Il est donc essentiel de s'assurer de la fiabilité et d'une évaluation rigoureuse des résultats rapportés, afin de pouvoir comparer efficacement les performances des différents modèles. Nous en discuterons plus en détail dans le chapitre 3.

7. Traitement de texte

Le traitement du langage naturel (NLP) fait référence à la branche de l'informatique, et plus particulièrement à la branche de l'intelligence artificielle (IA), qui vise à donner aux ordinateurs la capacité de comprendre les textes et les mots prononcés de la même manière que les êtres humains.

Le traitement du langage naturel combine la linguistique informatique (modélisation du langage humain basée sur des règles) avec des modèles statistiques, d'apprentissage

automatique et d'apprentissage en profondeur. Ensemble, ces technologies permettent aux ordinateurs de traiter le langage humain sous la forme d'un texte ou sous forme de données vocales et d'en « comprendre » le sens complet, avec l'intention et le ressenti du locuteur ou de l'auteur.

Le traitement du langage naturel fait fonctionner des programmes informatiques qui traduisent des textes d'une langue dans une autre, répondent à des commandes vocales et résument rapidement, voire en temps réel, de gros volumes de texte. Il y a de fortes chances que vous ayez eu affaire au traitement du langage naturel sous la forme de systèmes GPS à commande vocale, d'assistants numériques, de logiciels de dictée vocale, de chatbots de service à la clientèle et d'autres commodités pour les consommateurs. Mais le traitement du langage naturel joue également un rôle croissant dans les solutions d'entreprise qui permettent de simplifier les opérations commerciales, d'augmenter la productivité des employés et de simplifier les processus commerciaux essentiels [34].

Parmi les techniques les plus récentes pour la représentation et le traitement automatique du langage , on retrouve les Transformers.

7.1. Transformer

Les Transformers, et en particulier les modèles tels que SBERT (Sentence-BERT), représentent une avancée révolutionnaire dans le domaine du traitement automatique du langage naturel (NLP). Les Transformers sont des architectures de réseaux de neurones conçues pour gérer des tâches de séquences, permettant de capturer des relations complexes et contextuelles dans les données textuelles. SBERT, une variante de BERT (Bidirectional Encoder Representations from Transformers), est spécifiquement optimisé pour les comparaisons de phrases, offrant des capacités améliorées pour comprendre et analyser les similarités sémantiques entre différentes phrases

Partie2 : Représentation des données textuelle et volumique

8. La modélisation 3D

La modélisation 3D est le processus de création de représentations tridimensionnelles d'un objet ou d'une surface. Contrairement aux graphiques 2D traditionnels, qui sont limités à la hauteur et à la largeur, les graphiques 3D ajoutent la profondeur comme troisième dimension. Cette profondeur est obtenue en représentant les objets à l'aide de sommets, d'arêtes et de faces, qui forment collectivement des modèles 3D. Pour créer des graphiques 3D, des algorithmes mathématiques sont utilisés pour définir la forme, la texture, l'éclairage et le mouvement des objets dans un espace tridimensionnel virtuel. Ces algorithmes calculent et rendent les objets sur u affichage

ou un écran bidimensionnel, permettant aux utilisateurs de percevoir la profondeur et d'interagir avec l'environnement virtuel [36].

- **La représentation en nuage de points**

Un nuage de points est un ensemble non structuré de coordonnées 3D : P = {v1,...,vn}, où vi ∈ R3 stockées sous forme de tableaux ou listes, constituent la représentation la plus basique des données 3D. Généralement issus de techniques comme la structuration par mouvement, le SLAM ou le balayage laser, ils peuvent inclure des informations annexes telles que texture ou normales de surface. Leur absence de connectivité explicite entre points rend difficile l'approximation de la surface sous-jacente, nécessitant des méthodes d'estimation locale comme le calcul des normales par planarité sur les plus proches voisins. Si le rendu direct des nuages de points présente des avantages sur le rendu polygonal classique, leur manque de structure régulière en grille les rend peu adaptés en entrée des réseaux de neurones convolutionnels, d'où l'émergence de techniques dédiées comme PointNet pour le traitement permutation-invariant de ces données non structurées[37].

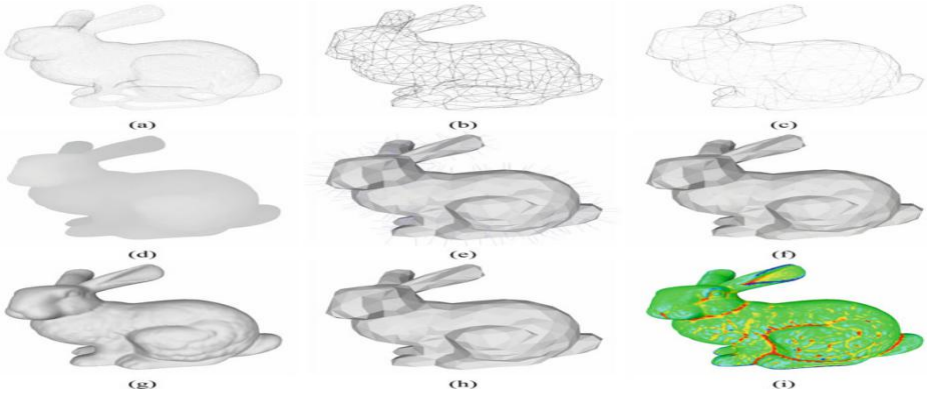


Figure 16 - Visualisation de différentes représentations 3D[38].

.

- **la représentation des maillages**

Un maillage est une représentation numérique discrète d'une surface 3D. Il se compose d'éléments géométriques simples interconnectés : des sommets qui sont des points 3D, des arêtes qui sont des segments reliant les sommets, et des faces qui sont généralement des triangles ou des quadrilatères formés par un ensemble de sommets. Plus formellement, un maillage triangulaire M est défini par un couple (V, K) où :

- V = {v1, v2, ..., vn} est l'ensemble des n sommets 3D
- K = {f1, f2, ..., fm} est l'ensemble des m faces triangulaires

Chaque face triangulaire fi est encodée par un triplet {j, k, l} d'indices faisant référence aux sommets vj, vk, vl de V qui la composent.

Un maillage quadrilatère suit une définition similaire, mais avec des faces composées de quatre sommets au lieu de trois. Cette représentation discrète par maillages s'est imposée comme standard dans de nombreuses applications 3D interactives. Elle offre une grande souplesse pour modéliser des formes complexes arbitraires, tout en permettant un rendu et des traitements géométriques efficaces sur le matériel graphique moderne. Sa nature purement discrète facilite également les opérations numériques comme le remaillage, le lissage ou la paramétrisation [39].

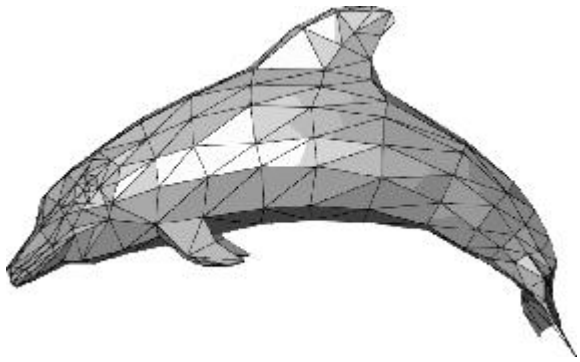


Figure 17- exemple sur la représentation des maillages [40].

- **la représentation voxel**

Une autre représentation importante des données 3D est La représentation par voxel fait référence à un pixel tridimensionnel (3D), où le terme « voxel » est dérivé de la combinaison de « volume » et de « pixel », représentant une valeur sur une grille régulière dans un espace 3D ,Cette représentation est couramment utilisée dans les applications d'infographie, d'imagerie médicale et de vision par ordinateur pour discrétiser l'espace 3D en petits éléments volumétriques.

Les grilles de voxels sont structurées dans un domaine euclidien, ce qui permet d'étendre facilement les techniques de traitement d'images 2D aux données 3D [41].

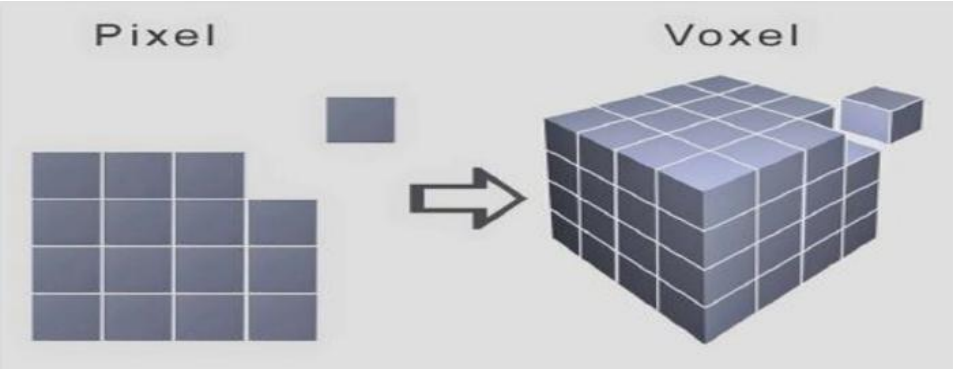


Figure 18- La représentation voxel est l'équivalent 3D de la représentation pixel en 2D, où un espace cubique est divisé en petits éléments de volume [42].

- La représentation par voxels consiste à discrétiser l'espace 3D en une grille régulière de petits cubes volumiques appelés voxels. Un objet 3D est encodé en marquant les voxels occupés par l'objet dans cette grille, tandis que les non marqués représentent l'espace vide. On peut voir la grille comme une fonction 3D discrète f(x,y,z) où chaque voxel (x,y,z) a une valeur binaire : 1 s'il fait partie de l'objet, 0 s'il représente le vide.
- Cependant, plutôt que des valeurs binaires, les représentations par voxels utilisent souvent des fonctions de distance signée tronquées (TSDF) pour encoder plus efficacement la surface. Une TSDF est une grille où chaque voxel stocke la distance signée à la surface la plus proche, tronquée à une valeur maximale pour limiter la mémoire. Cela permet une reconstruction lisse de la surface comme l'iso-surface à valeur 0.
- Contrairement aux nuages de points et maillages, la représentation voxel est ordonnée et régulière, similaire aux pixels 2D. Cette structure régulière facilite l'utilisation de filtres

convolutionnels dans les réseaux de neurones 3D. Cependant, elle requiert généralement plus de mémoire, bien que des techniques comme le hachage permettent de réduire ce coût.

- La représentation voxel offre ainsi plusieurs avantages : reconstruction robuste des surfaces, support des opérations booléennes, accélération du rendu et des traitements géométriques. Elle constitue une représentation 3D importante, complémentaire des autres comme les multi-vues ou RGB-D [43].

8.1.1. Architecture de SBERT

SBERT utilise une architecture Siamoise, qui comprend deux réseaux BERT identiques et parallèles. Chacun de ces réseaux BERT encode indépendamment une phrase d'entrée en une séquence de vecteurs de représentation de tokens. Cette architecture est inspirée des réseaux Siamois classiques utilisés pour l'apprentissage de similarités ,voici La figure 19 illustre le modèle BERT [44] .

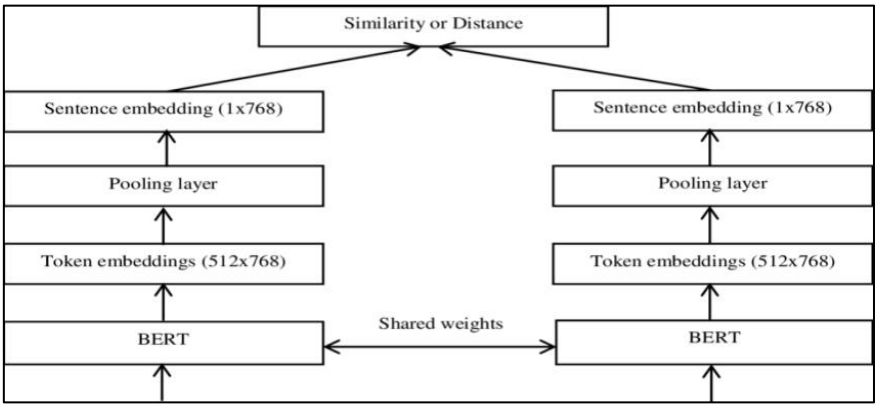


Figure 19- diagramme qui illustre une architecture ou un modèle impliquant BERT[45]

8.1.1.1. Mécanisme de Pooling

Après que les deux réseaux BERT ont encodé les phrases en séquences de vecteurs de représentation de tokens, une opération de pooling est appliquée pour obtenir un seul vecteur de représentation fixe pour chaque phrase.

Le pooling peut être effectué de deux manières principales :

Pooling par moyenne : Les vecteurs de représentation de tous les tokens de la phrase sont

moyennés pour produire un seul vecteur de représentation de phrase.

Max Pooling : Pour chaque dimension du vecteur de représentation, la valeur maximale parmi tous les vecteurs de tokens est sélectionnée, résultant en un vecteur de représentation de phrase unique [46].

8.1.1.2. Procédure d'entraînement par contrastive loss

SBERT est entraîné à l'aide d'une perte contrastive qui vise à rapprocher les représentations de phrases similaires dans l'espace vectoriel, tout en éloignant les représentations de phrases différentes. Cela est réalisé en optimisant une fonction de perte qui maximise la similarité cosinus entre les représentations de phrases similaires et minimise la similarité cosinus entre les représentations de phrases différentes.

La procédure d'entraînement consiste à échantillonner des paires de phrases (phrase1, phrase2) à partir des données d'entraînement, et à calculer leurs représentations respectives (u, v) à l'aide des réseaux BERT Siamois. La perte contrastive est alors calculée pour chaque paire et rétro-propagée pour mettre à jour les poids des réseaux [46].

Jeux de données d'entraînement

SBERT a été entraîné sur un mélange de données annotées manuellement et de données générées automatiquement :

- **Données annotées manuellement :** Des ensembles de données tels que la Stanford Natural Language Inference (SNLI) et la Multi-Genre Natural Language Inference (MNLI), où des annotations de similarité ont été fournies par des humains.
- **Données générées automatiquement :** Des paires de phrases ont été générées automatiquement à partir de différents corpus, comme Wikipedia, en extrayant des phrases voisines comme étant similaires, et des phrases éloignées comme étant différentes.

L'utilisation combinée de ces deux types de données d'entraînement a permis à SBERT d'apprendre des représentations de phrases robustes et généralisables.[46]

Partie 3: Travaux connexes

9. Travaux Connexes

On exposera plusieurs méthodologies distinctes permettant de passer d'une entrée textuelle à

une représentation 3D colorée :

9.1. Text to shape

A.Dans leur étude **"Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings" par Kevin Chen, Christopher B. Choy, Manolis Savva, Angel Chang, Thomas Funkhouser, Silvio Savarese**", [47] Kevin et al. présentent une méthode pour créer des formes 3D colorées à partir de descriptions en langage naturel. Leur approche se divise en deux grandes étapes :

- la récupération de texte à partir de formes (text-to-shape retrieval)
- la génération de formes à partir de texte (text-to-shape generation).

Tout d'abord, les auteurs développent un modèle permettant d'apprendre un espace commun de représentation pour les textes et les formes 3D colorées, à partir de descriptions en langage naturel. En utilisant un jeu de données inédit associant des descriptions textuelles à des formes 3D colorées, leur méthode repose sur les concepts d'apprentissage par association et d'apprentissage métrique. Cette approche aboutit à la création d'un espace d'intégration permettant de regrouper des formes et des descriptions similaires, établissant ainsi des liens sémantiques implicites. Ensuite, les auteurs présentent leur cadre de génération de formes à partir de texte. Contrairement à d'autres travaux sur la synthèse texte-image, Kevin et al. ne se basent pas sur des étiquettes catégorielles fines ni sur un pré-entraînement sur des ensembles de données volumineux. De plus, ils entraînent simultanément les composants d'encodage du texte et des formes, associant des points similaires au sein d'une même modalité (texte-texte ou forme-forme) et entre les deux modalités (texte-forme). La phase de récupération évalue la qualité de l'espace d'intégration texte-forme par rapport à des travaux antérieurs. Pour la génération texte-forme, les auteurs se concentrent sur la création de formes 3D colorées, étant donné que les descriptions de formes font souvent référence à des caractéristiques de couleur ou de matériau. Ils combinent le modèle d'intégration avec un nouveau cadre de Generative Adversarial Network (GAN) conditionnel de type Wasserstein, offrant des résultats de meilleure qualité et une plus grande diversité par rapport à une approche GAN conditionnelle

classique.Enfin, l'utilisation de l'arithmétique des vecteurs d'intégration et du générateur permet de manipuler les attributs des formes 3D de manière efficace, la figure suivante explique l'approche proposé de Kevin et al [47]

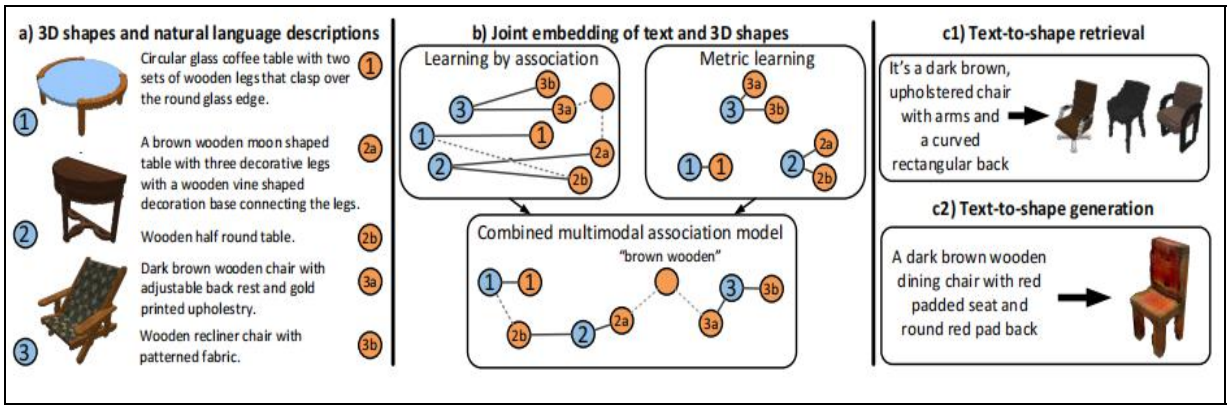


Figure 20- L'approche utilisée par Kevin Chen [47]

B.Dans leur etude "**Text-to-3D Shape Generation par Lee, H., Savva, M., & Chang, A.X.** " [48] L'article explore l'émergence de la génération de formes 3D à partir de texte, rendue possible par les progrès des représentations 3D, de l'apprentissage profond à grande échelle et des techniques de rendu différentiables. Il présente une analyse approfondie des méthodes sous-jacentes, classant les travaux récents selon le type de données de supervision utilisées. plusieurs architectures pour la génération de formes 3D à partir de texte:

1. Une architecture encodeur-décodeur entraînée sur le dataset ShapeNet. L'encodeur transforme les voxels 3D en une représentation latente. Un décodeur "occupancy network" utilise cette représentation pour reconstruire la forme 3D originale sous forme de grille d'occupation voxelisée.
2. Une architecture basée sur un modèle de flux normalisant (normalizing flow). Les représentations latentes issues de l'encodeur ShapeNet sont traitées conjointement avec des représentations visuelles CLIP obtenues à partir de rendus multi-vues des objets 3D. Le modèle de flux apprend à mapper ces représentations conjointes vers une distribution gaussienne. En inférence, des embeddings textuels CLIP sont utilisés pour obtenir des représentations latentes de formes, décodées ensuite par l'occupancy network pour générer la forme 3D finale.

3. Une architecture nommée CLIP-sculptor. Elle exploite des encodeurs d'images et de textes CLIP pré-entraints, mappés dans un espace latent commun. Un puissant modèle autorégressif est entraîné dans cet espace pour la reconstruction de formes 3D. Deux VAEs (un par résolution) encodent les voxels 3D en représentations latentes pour l'entraînement.

Ces techniques permettent aux modèles génératifs d'IA de créer du contenu 3D personnalisable à partir de descriptions textuelles. L'article discute des limites actuelles et propose des orientations futures pour ce domaine prometteur de la génération de formes contrôlée par le texte

Table 1- comparaison entre les travaux

Travail	Type	Coût	Pro
Text-to-3D Shape Generation	Text to Shape	Fait spécifiquement pour les formes libres uniquement	Produit des résultats de haute qualité avec détails fins pour les formes libres comme meubles, jouets, outils
Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings	Text to Shape	Limité à certaines catégories d'objets 3D	Résultats de haute qualité avec détails fins

9.2. Text to image to shape

C.Dans leur étude **"RealmDreamer: Text-Driven 3D Scene Generation with Inpainting and Depth Diffusion par Jaidev Shriram , Alex Trevithick, Lingjie Liu, Ravi**

Ramamoorthi “ [49] introduit une approche novatrice pour générer des scènes 3D détaillées et réalistes à partir de descriptions textuelles. Contrairement aux approches existantes limitées par le manque de données 3D, RealmDreamer exploite une représentation par "éclaboussures gaussiennes 3D" optimisée par l'intégration de trois éléments clés:

1. Générateurs texte-vers-image pour initialiser la scène 3D à partir du prompt textuel.
2. Modèles de diffusion en profondeur pour estimer les informations de profondeur et enrichir la structure géométrique 3D.
3. Diffusion conditionnelle d'images pour affiner la géométrie via une tâche d'inpainting 3D sur plusieurs vues.

Le processus commence par générer une image 2D initiale à partir du prompt textuel, puis la convertir en un nuage de points 3D grossier complété par inpainting 2D. La représentation 3D gaussienne initiale combine ce nuage de points et un volume d'occlusion calculé. Ensuite, un modèle d'inpainting 2D conditionné sur le texte complète les régions manquantes de la scène 3D rendue, guidé par une loss d'inpainting dans les espaces latent et image. Un modèle de diffusion prédit également des cartes de profondeur à partir des images complétées pour améliorer la géométrie 3D.La représentation 3D gaussienne est alors optimisée avec les losses d'inpainting et de profondeur combinées. Un réglage fin avec un générateur texte-vers-image, un lissage et une perte d'opacité améliorent la cohérence visuelle finale. Les résultats démontrent une nette supériorité de RealmDreamer pour générer des scènes 3D géométriquement cohérentes et visuellement riches, même sur des trajectoires de caméra complexes, sans données multi-vues. Cependant, les temps de calcul élevés et un possible flou dans les scènes encombrées restent à améliorer,la figure suivante explique l’approche RealmDreamer .

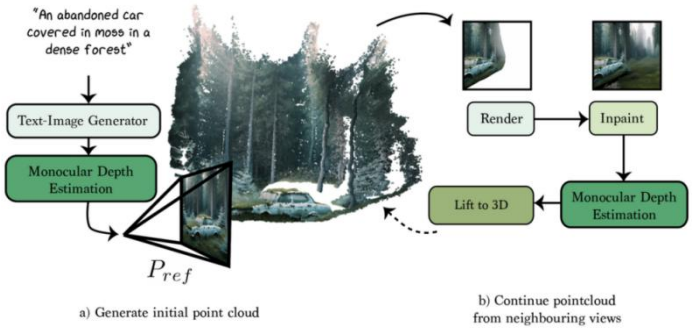


Figure 21- Initialisation de la représentation 3D par nuage de points à partir d'une description textuelle et de vues supplémentaires [49]

D. Dans leur étude **"Adversarial Synthesis of 3D Object Shape and Appearance from Sketches and Text' par Xinyi Yang, Peng Song, Chi Zhang et Hao Zhang"** [50] présentent une méthode pour créer des formes 3D réalistes en se basant sur ces entrées flexibles.à partir de deux types d'entrées simples : des croquis 2D et des descriptions textuelles,Les auteurs ont utilisé une architecture de type réseau génératif antagoniste (GAN), composée d'un :

Un générateur : Cet élément prend en entrée un croquis 2D et une description textuelle de l'objet, et produit en sortie :

- La forme 3D de l'objet
- Les textures et l'apparence de cet objet 3D

Un discriminateur : Cet élément reçoit en entrée une paire composée d'une forme 3D et de ses textures. Son rôle est de déterminer si cette paire provient des véritables objets 3D (les données réelles) ou si elle a été générée artificiellement par le générateur

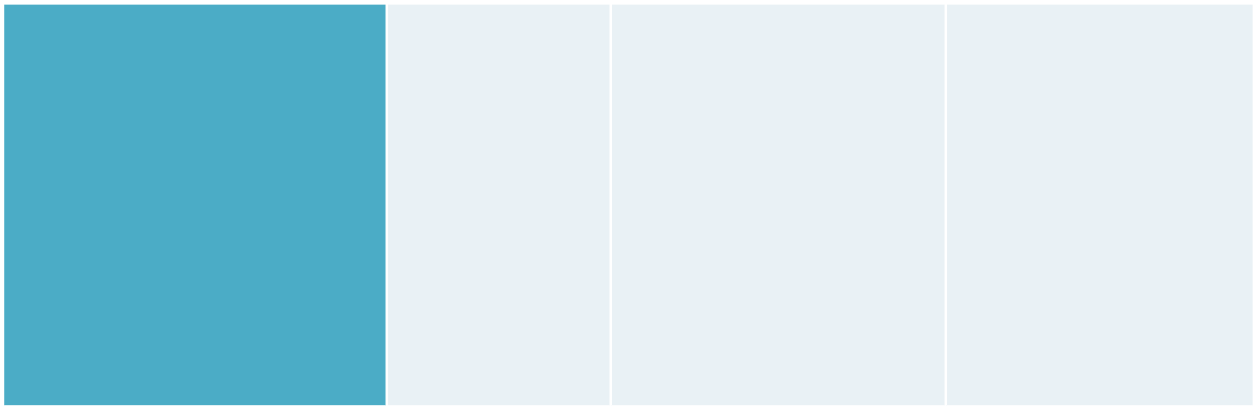
- Le système est entraîné de manière adversaire : le générateur cherche à tromper le discriminateur en générant des objets 3D et des textures très réalistes, tandis que le discriminateur apprend à mieux différencier les objets réels de ceux générés artificiellement. Cet entraînement permet d'améliorer progressivement les capacités du système à produire des résultats convaincants.
- Les auteurs ont entraîné leur modèle sur une base de données d'objets 3D comprenant des croquis 2D et des descriptions textuelles. Les résultats montrent que le système proposé est

capable de générer des formes 3D et des textures très réalistes à partir de ces entrées simples. De plus, le modèle peut également être utilisé pour créer des objets 3D à partir d'images réelles, en passant par une étape intermédiaire de transformation de l'image en croquis.

En conclusion, ce travail présente une approche innovante pour la génération automatique de modèles 3D d'objets, ouvrant la voie à de nouvelles possibilités pour la création d'objets 3D à partir d'entrées flexibles et accessibles (croquis et texte).

Table 2- comparaison entre les travaux

Travail	Type	Coût	Pro
RealmDreamer: Text-Driven 3D Scene Generation with Inpainting and Depth Diffusion	Text to Image to Shape	Nécessite beaucoup de ressources de calcul et de mémoire pour entraîner les grands modèles de diffusion - La génération peut être lente, surtout pour des scènes de grande taille	Génération de novo de scènes 3D photoréalistes à partir de descriptions textuelles, sans avoir besoin d'exemples 3D existants - Résultats visuellement réalistes et de haute qualité
Adversarial Synthesis of 3D Object Shape and Appearance from Sketches and Text	Text to Image to Shape	Limitation a la génération de formes simple	Interface intuitive utilisant croquis et texte - Génération de modèles 3D détaillés et texturés



10. Conclusion

Dans ce chapitre, nous nous sommes concentrés sur trois parties essentielles. Premièrement, nous avons abordé les notions fondamentales du machine learning et plus particulièrement du deep learning, ainsi que leurs algorithmes respectifs tels que les réseaux de neurones convolutionnels (CNN), les réseaux de neurones récurrents (RNN), les réseaux de neurones récurrents à mémoire courte et longue terme (LSTM), les auto-encodeurs et les réseaux antagonistes génératifs (GAN). Ensuite, nous avons exploré les différentes représentations liées à la modélisation 3D. Enfin, dans la section des travaux connexes, nous avons examiné deux approches principales : Text to Image to Shape et Text to 3D Shape, en détaillant les algorithmes sous-jacents. Cette étude approfondie nous a permis de mieux comprendre les enjeux et les défis actuels dans la génération de formes 3D à partir de descriptions textuelles. Dans le chapitre suivant, nous utiliserons le model CGAN (Conditional Generative Adversarial Networks) et l'Auto-encodeur pour notre architecture de génération de formes 3D, ouvrant ainsi la voie à de futures améliorations et applications innovantes dans ce domaine en pleine effervescence

Dans le chapitre suivant, nous allons explorer l'architecture de notre projet et les modèles qui

le composent. Nous allons générer des embeddings textuels avec BERT pour créer des formes 3D initiales à l'aide de deux modèles CGAN (Générateur Adversaire Conditionné) conditionnés par ces embeddings. Nous présenterons également le modèle CGAN avec Auto-Encoder et évaluerons les résultats de notre travail.

Chapitre 2:conception et réalisation

1. Introduction

Dans les chapitres précédents, nous avons vu les différent notion en machine learning et spécifiquement le deep learning et leur algorithme et des representations volumique et textuelle et travaux connexe. Maintenant, nous allons vous guider à travers notre propre approche et les modèles que nous avons choisis et développés . Tout d'abord, nous expliquerons comment nous avons généré les embeddings textuels en utilisant BERT de Google. Ensuite, nous expliquerons deux modèle CGAN (Réseau Génératif Adversaire Conditionné) où l'embedding textuel conditionne un vecteur aléatoire pour produire un tenseur PyTorch représentant la forme 3D initiale et autre méthode nous parlerons CGAN with Auto-Encoder qui affine cette forme initiale en l'encodant dans un vecteur latent que l'Auto-Encodeur décode pour reconstruire une forme 3D finale améliorée ce qui est l'objectif global de notre travail.

2. Architecture globale

Notre projet se compose de deux principales étapes pour obtenir à la fin des formes 3D.

- **conversion de description textuelle en vecteur embedding :**

Avant d'utiliser les descriptions textuelles dans un modèle quelconque, La première étape essentielle est de transformer ces informations en des formes numériques pour les utiliser avec des algorithmes d'apprentissage automatique: un processus appelé incorporation de texte. Les réseaux de neurones prennent des chiffres en entrée dans leur traitement et non pas du texte brut. Dans cette étape cruciale, le modèle pré-entraîné '***all-MiniLM-L6-v2***' a été utilisé. Il s'agit d'un modèle combinant les

approches de BERT et SBERT qui permet de convertir efficacement du texte en données numériques. Ce modèle prend en entrée une description textuelle et la transforme en un vecteur numérique dense, représenté sous forme de tableau NumPy. Dans la littérature, deux méthodes principales sont proposées pour réaliser cette conversion de texte en données numériques, mais l'approche du modèle '***all-MiniLM-L6-v2'*** est particulièrement adaptée. Cette représentation vectorielle de la description textuelle sera ensuite exploitée lors de l'étape suivante, qui consiste à générer les formes 3D correspondantes à partir de ces vecteurs

- **la génération des formes 3D correspondantes à partir des vecteurs numériques**

Dans cette deuxième étape, trois modèles sont utilisés pour créer des représentations 3D à partir de vecteurs numériques décrivant les descriptions textuelles : un modèle de génération adversaire conditionné (CGAN), un modèle combinant CGAN avec un auto-encodeur et un modèle de diffusion pour la génération de formes.

- Pour le modèle CGAN, un vecteur aléatoire est combiné avec l'embedding sémantique (la condition) du texte pour générer une forme 3D correspondant à la description textuelle.
- D'un autre côté, le modèle CGAN avec auto-encodeur utilise une architecture combinant un CGAN et un auto-encodeur pour améliorer la qualité de la forme 3D initiale.raffine et précise la forme 3D initiale générée par le CGAN, générant une forme 3D finale L'auto-encodeur comprend un encodeur avec des couches de convolution 3D qui réduisent progressivement la dimensionnalité de l'entrée, puis un décodeur avec des couches de convolution transposées 3D qui reconstituent la forme 3D à partir du vecteur latent encodé. Le CGAN est conditionné par l'embedding sémantique du texte et génère la forme 3D initiale, qui est ensuite affinée par l'auto-encodeur. La dernière couche de convolution du décodeur utilise une fonction d'activation sigmoïde pour produire des valeurs de voxels dans l'intervalle [0, 1], représentant la forme 3D finale.

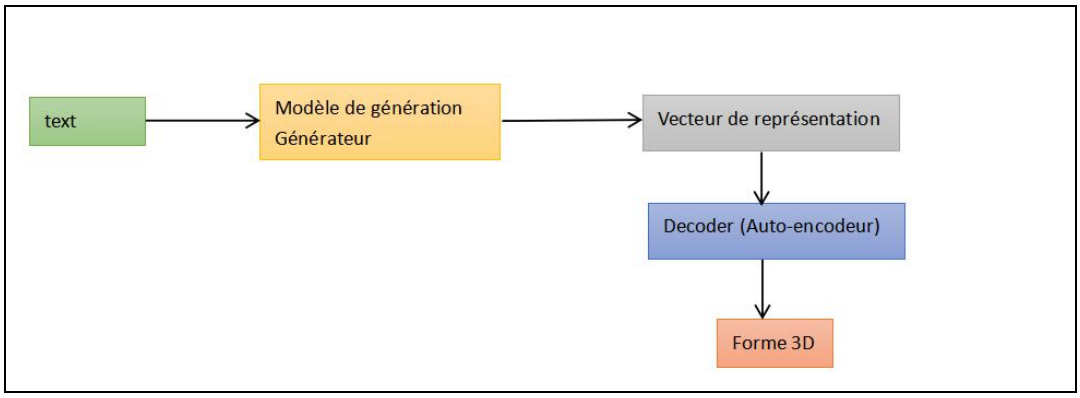


Figure 22- Architecture globale de travail

3. le modèle de plongement de texte all-MiniLM-L6-v2

le modèle de plongement de texte all-MiniLM-L6-v2 est un modèle de langage pré-entraîné, basé sur l'architecture MiniLM, qui a été développé pour les tâches de représentation sémantique du texte. Il s'agit d'une version compacte et efficace du modèle BERT, avec seulement 6 couches transformers au lieu de 12. ce modèle offre des performances élevées sur de nombreuses tâches de traitement du langage naturel, tout en étant plus léger et plus rapide que le modèle BERT original. Il est notamment performant pour les tâches de recherche de similitude sémantique entre phrases ou documents. Le modèle all-MiniLM-L6-v2 est particulièrement adapté aux applications nécessitant une représentation sémantique robuste du texte, tout en restant efficace en termes de ressources, comme la génération de texte, l'analyse de sentiments ou la classification de texte [51]

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1	65.98	52.83	59.41	4000	250 MB
all-MiniLM-L6-v2	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1	64.33	51.83	58.08	14200	80 MB

Figure 23 - Caractéristiques du modèle all-MiniLM-L6-V2 [51]

all-MiniLM-L6-v2	
Description:	All-round model tuned for many use-cases. Trained on a large and diverse dataset of over 1 billion training pairs.
Base Model:	nreimers/MiniLM-L6-H384-uncased
Max Sequence Length:	256
Dimensions:	384
Normalized Embeddings:	true
Suitable Score Functions:	dot-product (util.dot_score), cosine-similarity (util.cos_sim), euclidean distance
Size:	80 MB
Pooling:	Mean Pooling
Training Data:	1B+ training pairs. For details, see model card.
Model Card:	https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

Figure 24-Caractéristiques du modèle all-MiniLM-L6-V2 [51]

3.1. Avantages de l'utilisation du modèle all-MiniLM-L6-v2

1. Performances élevées : Le site indique que ce modèle offre des performances élevées sur de nombreuses tâches de traitement du langage naturel.
2. Compacité et efficacité : all-MiniLM-L6-v2 est une version compacte et efficace du modèle BERT, avec seulement 6 couches transformers au lieu de 12. Cela le rend plus léger et plus rapide que le modèle BERT original.
3. Robustesse des représentations sémantiques : Le modèle est particulièrement adapté aux applications nécessitant une représentation sémantique robuste du texte.
4. Polyvalence d'utilisation : all-MiniLM-L6-v2 peut être utilisé dans divers domaines tels que la génération de texte, l'analyse de sentiments, la classification de texte, etc.
5. Disponibilité et facilité d'utilisation : Le modèle est disponible gratuitement et peut être facilement chargé et utilisé via la bibliothèque Sentence Transformers, avec des exemples fournis.[51]

4. générer les formes 3D à partir des vecteurs de description textuelle

trois méthodes sont employées pour créer des formes 3D à partir de vecteurs de descriptions textuelles :

4.1. Méthode CGAN (Conditional Generative Adversarial Network)

4.1.1. Avantage de méthode CGAN

1.Généralisation améliorée

Les CGANs utilisent des techniques d'apprentissage profond pour combiner les informations conditionnelles avec les caractéristiques de la distribution de données d'entrée. Cela leur permet d'apprendre à générer des formes 3D plus réalistes et cohérentes, même pour des données non vues auparavant. Ils peuvent ainsi s'adapter à diverses variations et nuances dans les descriptions textuelles, améliorant ainsi leur capacité à généraliser.[52]

2.Contrôle conditionnel

La caractéristique clé des CGANs est d'intégrer un processus conditionnel, qui permet de contrôler la sortie de la génération de modèles 3D en fonction de diverses informations. Par exemple, on peut utiliser des étiquettes catégorielles, des vecteurs de caractéristiques ou d'autres types de données conditionnelles pour influencer la forme, la taille, les détails, le style ou d'autres aspects des modèles 3D générés. Cela offre une grande flexibilité et une capacité à créer une variété de formes 3D en utilisant des CGANs [53]

3. Flexibilité et adaptabilité

CGANs peuvent être adaptés et entraînés sur différents domaines et types de données, ce qui les rend polyvalents pour la génération de formes 3D à partir de descriptions textuelles dans divers contextes et applications.[54]

4.Qualité de génération améliorée

CGANs intègrent un processus d'apprentissage adversarial, ce qui les aide à générer des formes 3D de meilleure qualité et plus proches des modèles humains. Cela peut améliorer la précision et la fidélité des représentations 3D créées à partir de descriptions textuelles.[55]

5. Interopérabilité améliorée

En comparaison avec d'autres modèles de génération, les CGANs peuvent offrir une meilleure compréhension de l'influence des différents facteurs conditionnels sur la

génération des formes 3D. Cela signifie que les utilisateurs peuvent mieux comprendre comment les caractéristiques de la description textuelle influencent la sortie générée par le modèle, ce qui facilite l'analyse et l'optimisation du processus de génération.[56]

4.1.2. CGAN Avec Noise :

Les CGAN avec Noise sont une variante des CGAN standard où du bruit est ajouté aux entrées du générateur et du discriminateur, dans le but de rendre les réseaux plus robustes aux perturbations et d'améliorer leur capacité de généralisation. Le générateur, qui est un réseau de neurones artificiels, est conçu pour générer des images de résolution 32x32 avec 4 canaux, d'une taille totale de (4*32*32*32). Il prend en entrée un vecteur de bruit aléatoire de taille 384 et un vecteur d'embedding de texte de taille 384, auxquels du bruit gaussien est ajouté avant de les passer dans le reste du réseau. La structure du générateur est composée d'une séquence de couches linéaires, avec normalisation de batch et utilisation de la fonction d'activation Leaky ReLU. Il s'agit d'une classe qui hérite de la classe 'nn.Module' de la bibliothèque PyTorch de Python. De manière similaire, le discriminateur subit également l'ajout de bruit gaussien à ses entrées, et l'entraînement alternatif du générateur et du discriminateur, avec bruit, permet d'améliorer la robustesse et la généralisation du modèle Noisy CGAN dans son ensemble.

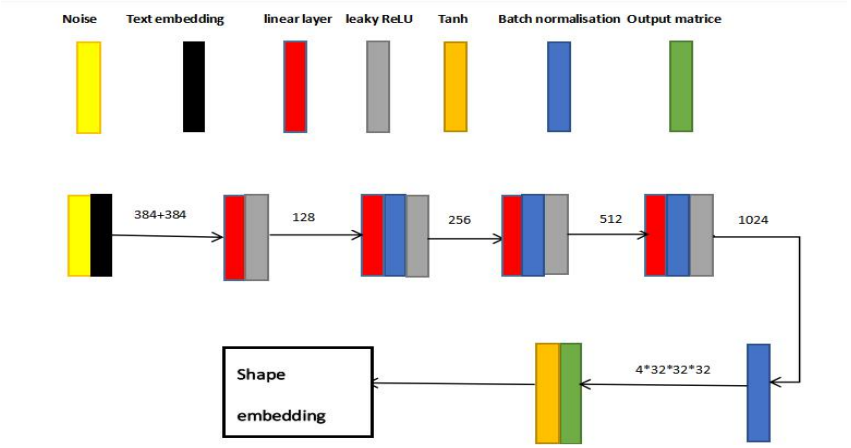


Figure 25- Architecture globale du générateur, CGAN avec noise

4.1.3. Structure de générateur

Couche d'entrée : Une couche linéaire de taille 384+384 prend en entrée la concaténation du vecteur de bruit et du vecteur d'embedding de texte, et produit une sortie de taille 128, suivie d'une activation Leaky ReLU.

Bloc 1 : Une couche linéaire de taille 128 génère un tenseur de taille 256, suivi d'une normalisation par lot avec un facteur de 0.8 et d'une activation Leaky ReLU.

Bloc 2 : Une couche linéaire de taille 256 génère un tenseur de taille 512, suivi d'une normalisation par lot et d'une activation Leaky ReLU.

Bloc 3 : Une couche linéaire de taille 512 génère un tenseur de taille 1024, suivi d'une normalisation par lot et d'une activation Leaky ReLU.

Couche de sortie : Une dernière couche linéaire de taille 1024 produit un tenseur de taille (4*32*32*32), sans normalisation par lot, mais en passant par une fonction d'activation Tanh.

4.1.4. Fonctionnement de générateur

Tout d'abord, le générateur combine le vecteur de bruit aléatoire (noise) avec l'encodage textuel (text embedding) en les concaténant. Cette opération fusionne les informations des deux entrées, permettant d'introduire à la fois de la variabilité grâce au bruit, et des caractéristiques spécifiques liées à la description textuelle dans le processus de génération.

Le vecteur combiné est ensuite transmis à travers les différentes couches du générateur. Chaque couche transforme progressivement les caractéristiques d'entrée et ajoute de la complexité à la sortie finale.

Le générateur utilise des couches entièrement connectées (fully connected layers) qui

effectuent une transformation linéaire en multipliant le vecteur d'entrée par des poids appris durant l'entraînement. Ces poids sont optimisés pour permettre la génération des images 3D souhaitées correspondant aux descriptions textuelles.

Des fonctions d'activation non-linéaires de type "LeakyReLU" sont appliquées pour introduire de la non-linéarité. Cela permet au générateur d'apprendre des relations complexes entre les caractéristiques d'entrée et la sortie désirée.

Une normalisation par lot (BatchNorm1d) est insérée après certaines couches afin de normaliser les activations. Cela accélère la convergence de l'apprentissage et stabilise l'entraînement du générateur.

Finalement, la dernière couche entièrement connectée est suivie d'une activation "Tanh" qui restreint la plage des valeurs de sortie entre -1 et 1. Ceci assure que l'image 3D générée aura des valeurs de voxels dans une plage utilisable.

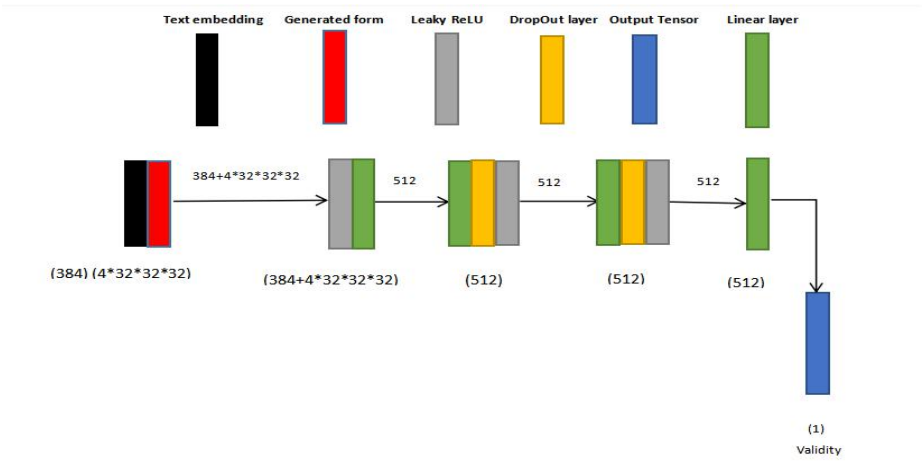


Figure 26- Architecture globale du discriminateur , CGAN avec noise

4.1.5. Structure de discriminateur

Couche d'entrée : Une couche linéaire de taille (latent_dim + (shape_size**3)*4) prend en entrée la concaténation de la forme 3D aplatie et de l'embedding de texte, et produit une sortie de taille 512, suivie d'une activation Leaky ReLU.

Bloc 1 : Une couche linéaire de taille 512 génère un tenseur de la même taille 512, suivi d'un dropout de 0.4 et d'une activation Leaky ReLU.

Bloc 2 : Une couche linéaire de taille 512 génère un tenseur de la même taille 512, suivi d'un dropout de 0.4 et d'une activation Leaky ReLU.

Couche de sortie : Une dernière couche linéaire de taille 512 produit un tenseur de taille 1, représentant le score de validité de l'entrée (réel ou généré).
Donc, le discriminateur prend en entrée la concaténation de la forme 3D et de l'embedding de texte, passe par deux blocs linéaires avec dropout, et produit finalement un score scalaire indiquant si l'entrée est réelle ou générée

4.1.6. Fonctionnement de discriminateur

Le discriminateur est défini comme une classe appelée "Discriminator. L'objectif du discriminateur est d'attribuer un score de "validité" ou de "réalisme" à une forme 3D générée, en la comparant à des formes 3D réelles. Le discriminateur est initialisé comme un réseau de neurones composé de plusieurs couches. La première couche est une couche linéaire qui projette l'entrée, qui est la concaténation de la forme 3D aplatie et de l'embedding textuel, dans un espace de dimension 512. Une activation non linéaire de type LeakyReLU est ensuite appliquée.Le tenseur résultant passe par deux blocs supplémentaires, chacun composé d'une couche linéaire de dimension 512, suivie d'un dropout avec un taux de 0.4 pour la régularisation, et d'une activation LeakyReLU. Ces blocs permettent d'extraire des caractéristiques de plus haut niveau à partir de l'entrée. Après ces trois blocs, le tenseur final de dimension 512 est projeté à travers une dernière couche linéaire pour produire une seule valeur scalaire. Cette valeur représente le score de "validité" ou de "réalisme" attribué par le discriminateur à l'entrée. La méthode "forward" du discriminateur prend en entrée une forme 3D et un embedding textuel. Elle commence par aplatir la forme 3D, puis concatène cette forme aplatie avec l'embedding textuel pour produire le vecteur d'entrée "d_in". Ce vecteur d'entrée est ensuite transmis au réseau de neurones séquentiel du discriminateur, qui produit en sortie le score de validité "validity",Le rôle du

discriminateur est d'apprendre à attribuer des scores élevés (proches de 1) aux formes 3D réelles, et des scores faibles (proches de 0) aux formes 3D générées de manière artificielle par le générateur. Cette capacité de discrimination est essentielle pour permettre au générateur d'apprendre à produire des formes 3D de plus en plus réalistes au fil de l'entraînement

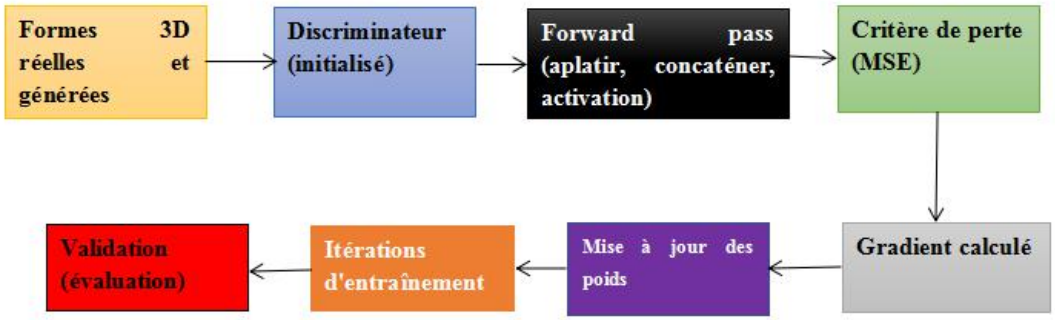


Figure 27- Schéma décrivant l'entraînement du discriminateur

4.1.7. La fonction de MSELoss

MSELoss (Mean Squared Error Loss) est la fonction de perte utilisée pour mesurer l'erreur entre les formes 3D générées par le modèle et les formes 3D cibles dans un contexte de génération de formes 3D à partir de descriptions textuelles avec une architecture GAN conditionnelle (CGAN). Elle est utilisée à la fois comme objectif pour le générateur et le discriminateur, avec des rôles distincts .

$$\text{MSELoss}(y\_pred, y\_true) = (1/n) * \Sigma(y\_pred[i] - y\_true[i])^2$$

- Pour le générateur, MSELoss évalue la précision de la forme 3D générée par rapport à la forme 3D cible, calculant la moyenne des carrés des erreurs entre les deux. Pour l'objectif est de minimiser la différence entre les sorties du discriminateur pour les formes 3D générées et la cible idéale de 1, pour générer des formes 3D qui sont perçues comme réalistes par le discriminateur.

Pour le discriminateur, MSELoss évalue la capacité du modèle à distinguer entre les formes 3D réelles et les formes 3D générées par le générateur, en calculant la moyenne des carrés des erreurs entre les sorties du discriminateur et les étiquettes de sortie attendues. L'objectif est de minimiser la différence entre les sorties du discriminateur pour les formes 3D réelles (cible 1) et celles pour les formes 3D générées (cible 0), afin de permettre au modèle de classer correctement les formes 3D réelles comme étant réelles et les formes générées comme étant fausses.

L'utilisation de la MSELoss dans ce CGAN fournit un signal d'erreur différentiable pour la mise à jour des poids du générateur et du discriminateur pendant l'entraînement. Cependant, d'autres fonctions de perte comme la cross-entropy binaire sont plus couramment utilisées dans les GANs, car elles tendent à converger plus rapidement et de manière plus stable.

4.2. Méthode CGAN avec Auto-Encodeur

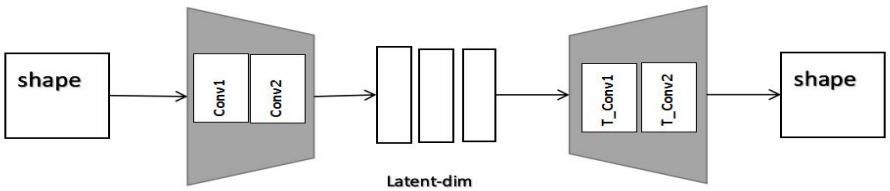


Figure 28-Architecture globale du l'Auto-encodeur

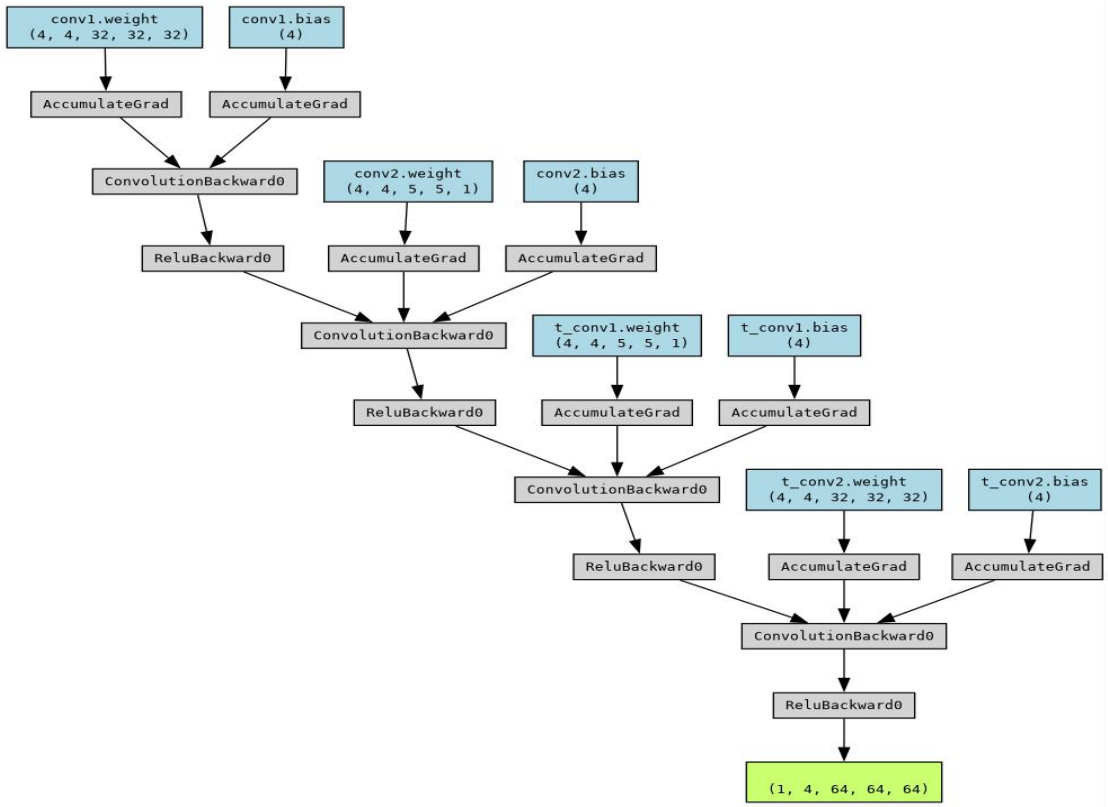


Figure 29- Architecture intérieure de l'Auto-encodeur

4.2.1. Architecture du CGAN avec Auto-Encodeur

Le CGAN avec l'auto-encodeur est composé de trois parties :

- **Générateur (G) :**

Le générateur G prend un vecteur de bruit aléatoire z concaténé avec un embedding textuel décrivant la forme 3D souhaitée. Ce vecteur d'entrée passe d'abord par une couche linéaire, une non-linéarité LeakyReLU et une normalisation BatchNorm1d pour extraire des caractéristiques de bas niveau. Ensuite, il traverse plusieurs blocs composés chacun d'une couche linéaire, d'un dropout et d'une LeakyReLU, permettant d'extraire progressivement des caractéristiques de plus haut niveau. Finalement, une dernière couche linéaire suivie d'une fonction Tanh produit la forme 3D générée synthétiquement, dont la taille correspond à la sortie désirée (ex: 64x64x64 voxels). Durant l'entraînement, les poids de ces couches sont mis à jour pour que le discriminateur D juge les formes générées comme de plus en plus réalistes

❖ **Auto-Encodeur (AE)**

L'auto-encoder est un modèle de réseaux de neurones qui prend la forme synthétique générée par le générateur comme entrée, encode la forme en une représentation compressée (z), puis décode la représentation compressée pour obtenir la forme reconstruite. L'auto-encoder est entraîné pour reconstruire les formes synthétiques générées par le générateur

Nous allons examiner en détail la fonctionnalité de l'encodeur et du décodeur, en éclairant leur rôle dans notre architecture.

❖ **Encodeur**

L'encodeur vise à compresser la forme 3D d'entrée générée par G dans une représentation latente de dimension réduite, via deux couches convolutives 3D successives. La première ("conv1") applique des noyaux 3D de taille (4, 4, 32, 32, 32) avec un biais pour extraire des caractéristiques de bas niveau de la forme, suivie d'une non-linéarité ReLU dont les gradients sont calculés ("ReluBackward0") pendant la rétropropagation. Ensuite, la seconde couche ("conv2") utilise des noyaux (4, 4, 5, 5, 1) et un biais pour combiner les caractéristiques précédentes et encoder la forme en une représentation latente compressée de plus haut niveau, à nouveau suivie d'une ReLU avec calcul des gradients. Cette sortie finale de "conv2" constitue donc l'encodage condensé capturant l'essentiel de la forme 3D initiale.

❖ **Décodeur**

Le décodeur prend la représentation latente compressée provenant de l'encodeur et la décode pour reconstruire une forme 3D de même taille que l'entrée originale de G, via deux couches de convolution transposée 3D. La première ("t_conv1") utilise des noyaux (4, 4, 5, 5, 1) avec un biais pour commencer à reconstruire grossièrement la forme à partir du code latent, suivie d'une non-linéarité ReLU avec calcul des gradients ("ReluBackward0"). Ensuite, la seconde couche ("t_conv2") applique des noyaux (4, 4, 32, 32, 32) et un biais pour combiner les caractéristiques décodées par "t_conv1" et reconstruire la forme 3D finale, à nouveau suivie d'une ReLU. Enfin, une

dernière activation comme la sigmoïde restreint les valeurs pour obtenir la reconstruction (1, 4, 64, 64, 64). Durant l'entraînement, les gradients des poids sont calculés par "ConvolutionBackward0" et "AccumulateGrad" afin de minimiser l'erreur entre la sortie du décodeur et la forme 3D d'entrée

4.2.2. Entraînement

L'entraînement du GAN avec auto-encodeur consiste à optimiser les paramètres du générateur et de l'auto-encodeur pour minimiser la perte totale. La perte totale est calculée comme suit :

- **Perte du générateur (G) :** La perte du générateur est calculée a l'aide de la fonction La fonction de perte MSE (Mean Squared Error) est utilisée pour entraîner le Générateur, qui mesure la différence entre la forme générée et une forme réelle comme la différence entre les prédictions du discriminateur et la vérité terrain. Pour les formes générées, la vérité terrain est 1. La perte du générateur mesure la capacité du générateur à produire des formes 3D qui peuvent tromper le discriminateur en faisant croire qu'elles sont réelles
- **Perte de l'auto-encodeur (AE) :** La perte de l'auto-encodeur est calculée a l'aide de la fonction de perte MSE (Mean Squared Error) comme la différence entre les formes générées par le générateur et les formes reconstruites par l'auto-encodeur. Cette perte mesure la capacité de l'auto-encodeur à reconstruire les formes 3D générées avec précision. La perte de l'auto-encodeur est minimisée lorsque l'auto-encodeur réussit à reconstruire les formes 3D avec une erreur minimale.

5. Conclusion

Dans ce chapitre, nous exposons en détails l'architecture globale que nous avons conçue et développée dans le but de générer des formes 3D à partir de descriptions textuelles. Nous débuterons par présenter le modèle SBERT utilisé pour encoder les textes en représentations vectorielles. Ensuite, nous détaillerons les trois approches majeures employées pour la génération des formes 3D : les réseaux antagonistes génératifs conditionnels (CGAN) avec injection de bruit, CGAN avec l'auto-encodeurs, Nous expliquerons le fonctionnement de chacune de ces méthodes et leur rôle spécifique dans le processus de génération des formes 3D correspondant aux descriptions textuelles fournies en entrée

Le chapitre suivant sera consacré à la présentation des différents outils utilisés dans notre processus, ainsi qu'à une évaluation détaillée de notre travail.

Chapitre 3 : Test et résultat

1. Introduction

Le chapitre précédent présenter en détail l'architecture des différents modèles que nous avons développés. Dans la continuité, ce chapitre sera consacré au processus de mise en œuvre de ces modèles. Nous débuterons par une description des outils utilisés pour leur construction. Ensuite, nous aborderons les différents paramètres qui ont été définis lors de leur entraînement. Enfin, nous fournirons un aperçu synthétique de l'évaluation réalisée afin d'analyser les performances de nos modèles.

2. Les outils utilisés

2.1. Google collabotary

Google Colab (abréviation de "Colaboratory") est un service cloud gratuit proposé par Google, basé sur Jupyter Notebook et destiné à la formation et à la recherche en apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud, sans avoir besoin d'installer de software sur notre ordinateur, à l'exception d'un navigateur. Il s'agit d'un service hébergé de notebooks Jupyter, qui n'exige aucune configuration et offre un accès gratuit aux ressources informatiques, notamment des GPU (Graphic Processing Units) pour améliorer les performances en deep learning et autres applications [57]

colab permet :

- D’améliorer vos compétences de codage en langage de programmation Python.
- De développer des applications en Deep Learning en utilisant des bibliothèques Python populaires telles que Keras, TensorFlow, PyTorch et OpenCV.
- D’utiliser un environnement de développement (Jupyter Notebook) qui ne nécessite

aucune configuration.

- Développement et entraînement de réseaux de neurone

2.2. Python

Python, un langage de programmation créé en 1989 par Guido van Rossum aux Pays-Bas, a été nommé en hommage à la série télévisée Monty Python's Flying Circus, dont van Rossum est fan. La première version publique de Python a été publiée en 1991. Ce langage de niveau supérieur est connu pour sa simplicité d'écriture et de lecture, rendant son apprentissage facile, et est considéré comme un langage de programmation ouvert. Python est polyvalent et utilisé dans de nombreux domaines, tels que la création d'applications informatiques avec des interfaces utilisateur, le développement de programmes web et le contrôle des performances des programmes les plus populaires. Python convient pour programmer des projets simples pour les débutants et pour réaliser des projets complexes, tout comme d'autres langages de programmation. Il est souvent recommandé aux débutants en programmation d'apprendre Python car c'est l'un des langages de logiciel d'apprentissage les plus rapides à apprendre[58].

2.3. NumPy

NumPy (Numerical Python) est une bibliothèque open source pour le langage Python, qui ajoute un support pour les vecteurs et matrices multidimensionnelles ainsi que de nombreuses fonctions mathématiques permettant d'effectuer des opérations sur ces vecteurs et matrices. NumPy est devenu la bibliothèque de référence pour le calcul scientifique et l'analyse de données en Python. Elle sert de fondation pour d'autres bibliothèques Python telles que Pandas, SciPy et Matplotlib.

NumPy a été créé en 2005 par Travis Oliphant, en s'appuyant sur les précédents travaux des bibliothèques Numeric et Numarray. Elle offre des structures de données performantes ainsi qu'un grand nombre d'opérations numériques optimisées [59].

2.4. NRRD

NRRD est une bibliothèque et un format de fichier pour la représentation et le traitement de données raster à n dimensions. Il a été développé par Gordon Kindlmann pour prendre en charge les applications de visualisation scientifique et de traitement d'images. NRRD peut être consulté et modifié via la bibliothèque Python Pynrrd [60].

2.5. BERT

BERT (Bidirectional Encoder Representations from Transformer) est un modèle de langage profond et transformé conçu pour la compréhension et la génération de texte. BERT est un acronyme pour "Bidirectional Encoder Representations from Transformer". Ce modèle est particulièrement innovant car il est entraîné sur des tâches de pré-entraînement telles que la classification de segments, la prédiction de mots manquants et la prédiction de segments, ce qui permet de générer des représentations contextuelles efficaces et puissantes pour le texte. Grâce à cette approche, BERT est capable de capturer les relations entre les mots dans un contexte bidirectionnel, ce qui améliore considérablement ses performances dans diverses tâches de traitement du langage naturel, telles que la classification de documents, la traduction machine-à-machine, la question-réponse et la génération de texte [61].

2.6. PyTorch

PyTorch est une bibliothèque d'intelligence artificielle créée par Meta, écrite en Python. Elle permet le développement de deep learning (apprentissage profond) et le développement de réseaux de neurones artificiels. En utilisant plusieurs variables, vous pouvez effectuer des calculs de gradients ou utiliser des tableaux multidimensionnels grâce aux tenseurs. Depuis 2016, PyTorch est disponible en open source sous la licence BSD modifiée. En 2018, la bibliothèque de Meta a été fusionnée avec Caffe2, une infrastructure d'apprentissage en profondeur adaptée au déploiement. Caffe2 est capable de gérer des algorithmes d'apprentissage avec des

dizaines de milliards de paramètres[62].

2.7. Kaggle

une communauté en ligne pour les experts en données et les apprenants de machines, qui fait partie de Google LLC. Kaggle a commencé en organisant des compétitions en intelligence artificielle et a étendu ses offres pour inclure une plateforme de données publiques, un espace de travail cloud pour les sciences de données et des ressources d'apprentissage en intelligence artificielle. Kaggle Kernels, un environnement de calcul en nuage, permet la réalisation d'analyses reproductibles et collaboratives. Il prend en charge les scripts en R, Python, Jupyter Notebooks et rapports RMarkdown. Kaggle Kernels offre des ressources de calcul en nuage gratuites, équipées d'un GPU Nvidia K80 et 16 Go de RAM, avec une durée de session limitée à 6 heures. Notre travail actuel a été influencé par un défi Kaggle annoncé récemment en 2018 [63].

2.8. Tkinter

Tkinter est la bibliothèque standard de Python pour le développement d'interfaces graphiques utilisateur (GUI). Elle met à disposition un ensemble complet d'outils et de composants visuels, appelés widgets, permettant de construire des applications de bureau dotées d'une interface graphique conviviale. Un avantage clé de Tkinter est qu'elle est intégrée à la plupart des installations Python, la rendant ainsi immédiatement accessible aux développeurs souhaitant créer des applications GUI, sans avoir besoin d'installer des bibliothèques tierces supplémentaires. Cette inclusion native dans Python facilite grandement la conception d'interfaces utilisateur graphiques pour enrichir les programmes [64].

2.9. Visual Studio Code

Est un éditeur de code source léger mais puissant qui prend en charge de nombreux langages de programmation et offre une personnalisation flexible et des fonctionnalités de logiciels de qualité [65].

2.10. Matplotlib

est une bibliothèque Python complète pour la visualisation de données, permettant de créer des graphiques statiques, animés ou interactifs de haute qualité. Elle offre de nombreuses options de personnalisation de l'apparence visuelle et de la mise en page des graphes, ainsi que la possibilité de les exporter dans divers formats et de les intégrer facilement à JupyterLab ou des interfaces graphiques. Que les besoins soient simples ou complexes, Matplotlib propose des fonctionnalités riches et flexibles. De plus, elle bénéficie d'un vaste écosystème de packages tiers complémentaires, renforçant encore ses capacités[66].

3. Dataset utilisé

Nous avons utilisé deux ensembles de données pour mener à bien notre projet. Les voici :

3.1. Shapenet :

ShapeNet est un vaste référentiel de modèles 3D riche en informations. Il contient des modèles couvrant une multitude de catégories sémantiques. Contrairement aux référentiels de modèles 3D précédents, ShapeNet fournit des ensembles étendus d'annotations pour chaque modèle et des liens entre les modèles du référentiel et d'autres données multimédias externes. Tout comme ImageNet, ShapeNet offre une vue hiérarchique catégorisée des données selon les synsets de WordNet. Contrairement aux autres référentiels de modèles, ShapeNet fournit également un riche ensemble d'annotations pour chaque forme et des correspondances entre les formes. Les annotations incluent des attributs géométriques tels que les vecteurs d'orientation verticaux et avant, les parties et points clés, les symétries de forme et l'échelle de l'objet en unités réelles. 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 [67].

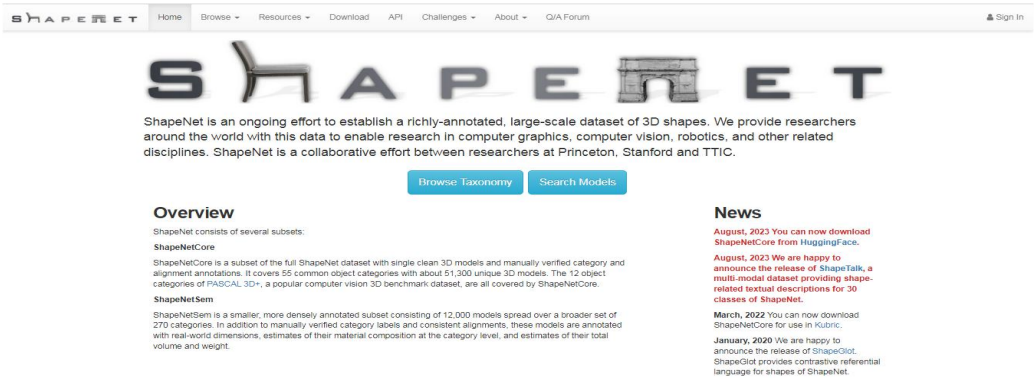


Figure 30- Page d'accueil Shapenet[67].

un exemple du dataset Shapenet est montre dans la figure suivante :

Figure 31- Exemple de dataset ShapeNet [67].

3.2. Primitives :

Afin de pouvoir évaluer quantitativement notre modèle de manière systématique, nous avons utilisé un ensemble de données comprenant des primitives géométriques 3D ainsi que leurs descriptions textuelles correspondantes. Cet ensemble a été généré en voxélisant 6 types de primitives (cuboïdes, ellipsoïdes, cylindres, cônes, pyramides et tores) déclinées en 14 couleurs différentes et 9 tailles variées. Les variations de couleur et de taille ont été soumises à des perturbations aléatoires, produisant ainsi 10 échantillons pour chacune des 756 configurations primitives possibles, soit un total de 7560 formes voxélisées. Nous avons ensuite créé les descriptions textuelles associées en utilisant une approche par modèles permettant d'insérer les mots décrivant la forme, la taille et la couleur dans différents ordres, générant des phrases telles que "un grand cylindre rouge est étroit et haut" (voir Annexe B pour plus de détails). Au final, 192 602 descriptions ont été produites, soit une moyenne d'environ 25 descriptions par

configuration primitive. Bien que ce texte synthétique ne corresponde pas au langage naturel, il permet de disposer d'un benchmark clair avec une correspondance évidente entre les descriptions et les attributs de chaque forme primitive.un exemple du dataset Primitive est montre dans la figure 30.

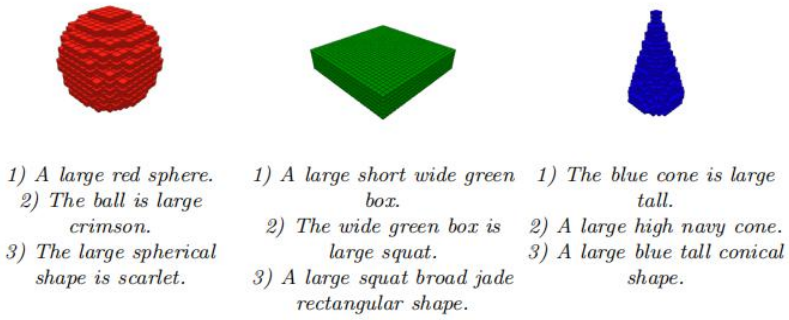


Figure 32- Exemple de la dataset Primitive[67].

La figure suivante présente une comparaison détaillée des caractéristiques des deux datasets :

	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 33- Statistiques comparatives des datasets ShapeNet et Primitive[68].

4. métriques d'évaluation du modèle

Dans cette partie, nous abordons la façon d'évaluer les performances de notre modèle, ce qui en distingue le meilleur des autres, ainsi que la prédiction de ses bons résultats en test et développement. Un large éventail d'indices d'évaluation a été suggéré pour évaluer la qualité de la production de classification.

4.1. Évaluer la performance des algorithmes de classification

pour les problèmes de classification, l'évaluation des performances est plus complexe que pour la régression numérique. Une technique d'évaluation clé est la matrice de confusion, qui permet de visualiser clairement où l'algorithme commet des erreurs de classification en comparant les prédictions aux véritables classes. Un pourcentage élevé d'individus mal classés dans cette matrice remet en cause les capacités prédictives de l'algorithme. Les lignes montrent les classes prédites par l'algorithme, tandis que les colonnes indiquent les classes réelles des données.

L'examen de la matrice de confusion donne ainsi un aperçu rapide des faiblesses potentielles d'un algorithme de classification et permet de quantifier son taux d'erreurs, facilitant l'évaluation et l'interprétation de ses performances [69].

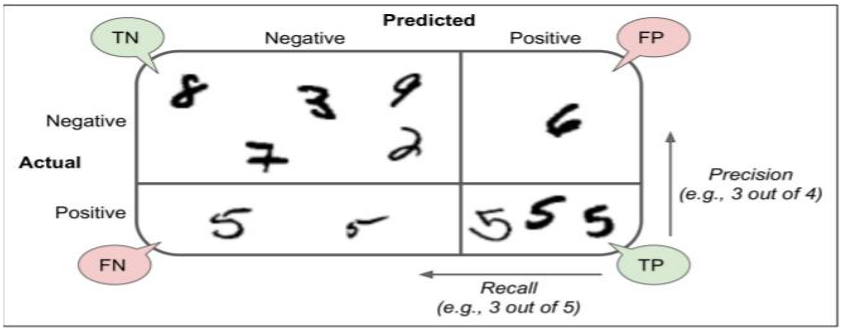


Figure 34-Une matrice de confusion illustrée [69].

L'évaluation des performances expérimentales mesurait la précision, le rappel et le score F1. Méthodes pour les performances de mesure sont les suivantes.

- **Le taux de succès (accuracy)**

le taux de succès fait généralement référence à la proportion globale de prédictions correctes par rapport au total des instances. C'est un ratio entre le nombre de prédictions correctes (vrais positifs + vrais négatifs) et le nombre total d'instances[69].

- **La précision (precision)**

La précision mesure la proportion de prédictions positives qui sont effectivement correctes (vrais positifs) parmi toutes les prédictions positives faites par le modèle. Elle indique à quel point le classifieur est précis lorsqu'il prédit la classe positive[69].

TP / (TP + FP)

- **Le rappel (recall) :**

Le rappel représente la proportion d'instances positives réelles qui sont correctement identifiées par le modèle (vrais positifs). Il reflète la capacité du classifieur à détecter tous les cas positifs, sans en manquer.[69]

TP/(TP + FN)

- **Le score F1**

le score F1 offre un équilibre entre la précision (proportion de prédictions positives correctes) et le rappel (proportion de vrais positifs détectés). Il atteint sa valeur optimale de 1 lorsque précision et rappel sont maximaux, et une valeur minimale de 0 dans le pire des cas. Contrairement à la précision seule, le F1 tient compte à la fois des faux positifs et des faux négatifs. Bien qu'un peu moins intuitif que la précision globale, le F1 est souvent plus pertinent, surtout avec des classes déséquilibrées. Alors que la précision suppose que les coûts des erreurs sont équivalents, le F1 permet d'accorder plus d'importance relative à la précision ou au rappel si leurs coûts diffèrent fortement. C'est une métrique unique qui synthétise précision et rappel en une seule valeur harmonique[70].

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

• Faux Positif (FP) : Nombre de données anormales incorrectement classées comme normales.
•Vrai Négatif (TN) : Nombre de données anormales correctement classées comme anormales.
•Faux Négatif (FN) : Nombre de données normales incorrectement classées comme anormales.

5. Expérience

Dans cette partie, nous passerons en revue les différentes architectures employées dans notre travail afin d'identifier celles qui ont donné les meilleurs résultats.

5.1. CGAN

Dans les réseaux génératifs conditionnels, le générateur crée des formes 3D en

combinant les embeddings (représentations vectorielles) de descriptions textuelles avec un vecteur de bruit aléatoire. L'ajout de cette composante aléatoire permet d'obtenir une plus grande variété de formes générées. Cette approche, qui fusionne les informations textuelles et un élément de hasard, vise à améliorer à la fois la qualité et la diversité des modèles 3D produits par le système**.**

Voici Les paramètres utilisés pour l'entraînement du modèle CGAN avec noise sont les suivants :

Nombre d‟epochs : 200
Batch size : 32
Nombre de worker = 4
Learning rate = 0.0002
Latent vector = 384
Shape_size = 32
Nombre de channels = 4

- Voici un schéma de l'évolution de la fonction de perte au cours de l'entraînement du modèle CGAN sur 400 époques, tracé tous les 20 époques pour le jeu de données ShapeNet.

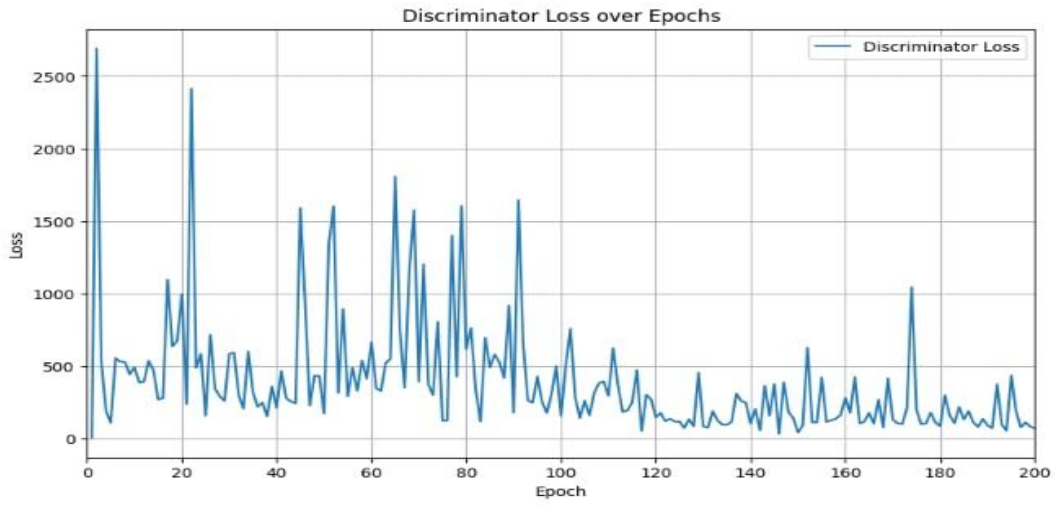


Figure 35- la fonction de perte du discriminateur pour ShapeNet

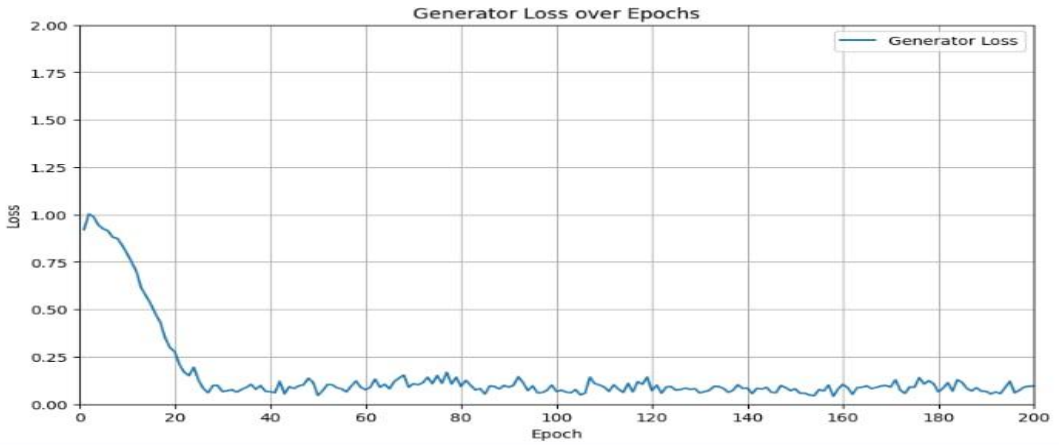


Figure 36- la fonction de perte du Generateur pour ShapeNet

5.2. CGAN avec Auto-Encodeur

Nous avons utilisé l'auto-encodeur pour apprendre les plongements de forme, voici les paramètres que nous avons utilisés pour entraîner le modèle :

Nombre d‟epochs : 100
Batch size : 32
Nombre de worker = 4
Learning rate = 0.0002
Latent vector = 384
Shape_size = 32
Nombre de channels = 4

- Voici un schéma de l'évolution de la fonction de perte au cours de l'entraînement du modèle Auto-encodeur sur 100 époques, tracé tous les 20 époques pour la dataset ShapeNet.

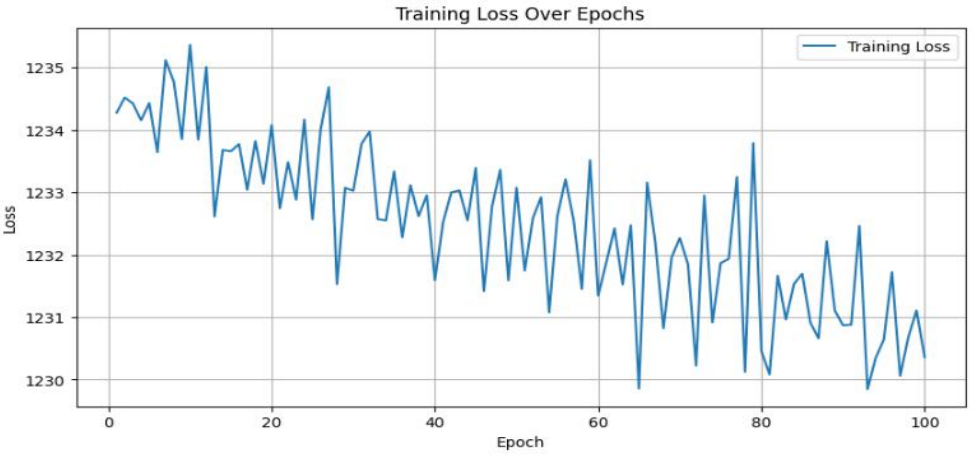


Figure 37- La fonction de perte de l'Auto-encodeur.

6. Notre interface

Notre inteface est développée avec Tkinter, Elle offre aux utilisateurs la possibilité de saisir des descriptions, de choisir un modèle de génération, et de visualiser les résultats, simplifiant ainsi la création de formes 3D pour les non-experts.

- **Fenetre "3D Shape Generation" :**

Ce panneau principal (Figure 36) présente deux options :

- "Convert text to shape" : permet de lancer le processus de conversion de texte en forme 3D.
- "Display from dataset" : offre probablement la possibilité de visualiser des formes existantes dans une base de données.

- **Fenetre "Choose a Model" :**

Cette section (figure 37) permet à l'utilisateur de choisir entre deux modèles de conversion :

- "CGAN with Noise"
- "CGAN with Auto-Encoder" Ces options déterminent l'algorithme utilisé pour la génération de forme.

- **Fenetre "Conversion" :**

Ce panneau est l'interface de saisie pour l'utilisateur. Il comprend :

- Un champ de texte pour entrer la description de la forme souhaitée.
- Un bouton "Generate Shape" pour lancer la conversion.
- Une option "Back" pour revenir à l'écran précédent.

- **Fenetre "3D form" :**

Ce panneau affiche le fichier nrrd qui contient la forme dont le model vient de générer

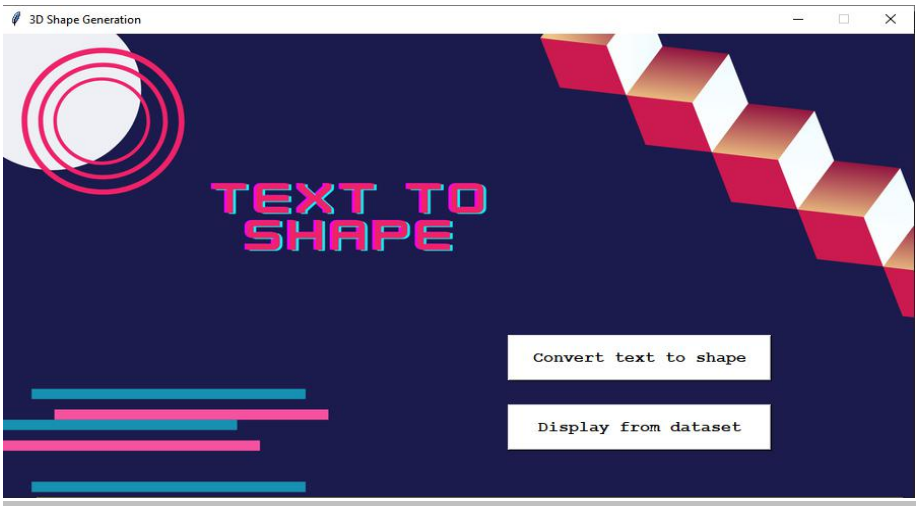


Figure 38 - La fenêtre principale de notre interface

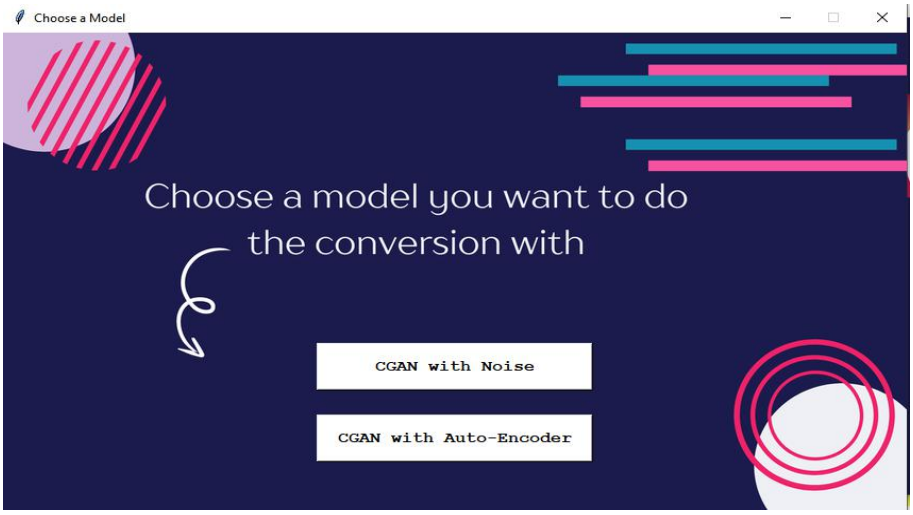


Figure 39 - La fenêtre de choix de modèle

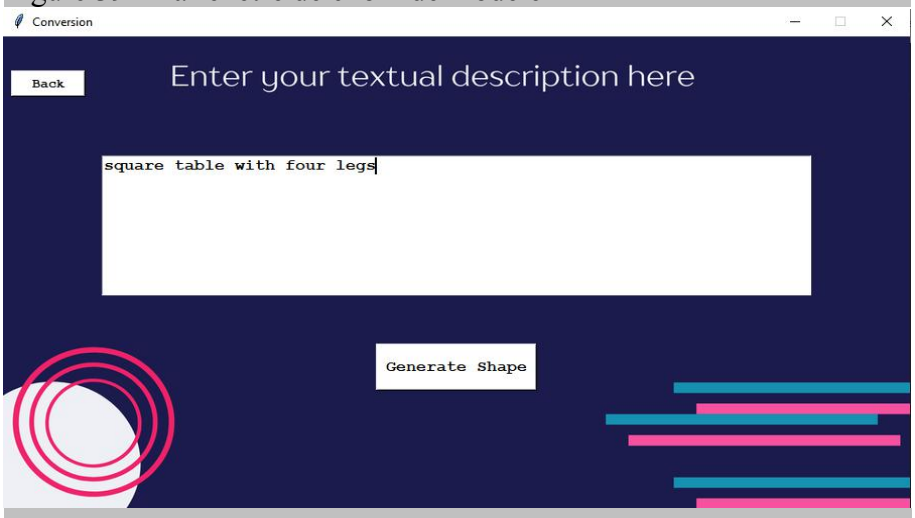


Figure 40 - La fenêtre de conversion de notre interface

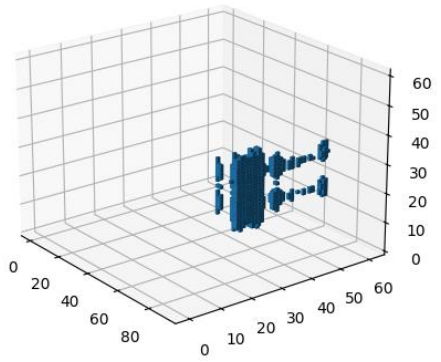


Figure 41 -Affichage d'un fichier NRRD généré par l'un de no modéles

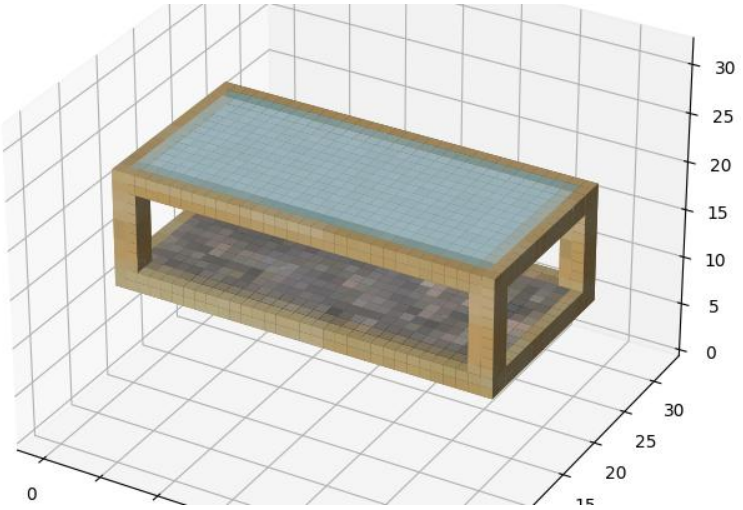


Figure 42 - Affichage d'un fichier NRRD d'une table

7. Analyse comparative

Dans notre étude, nous présentons des résultats des métriques d'évaluation pour la datasets ShapeNet.

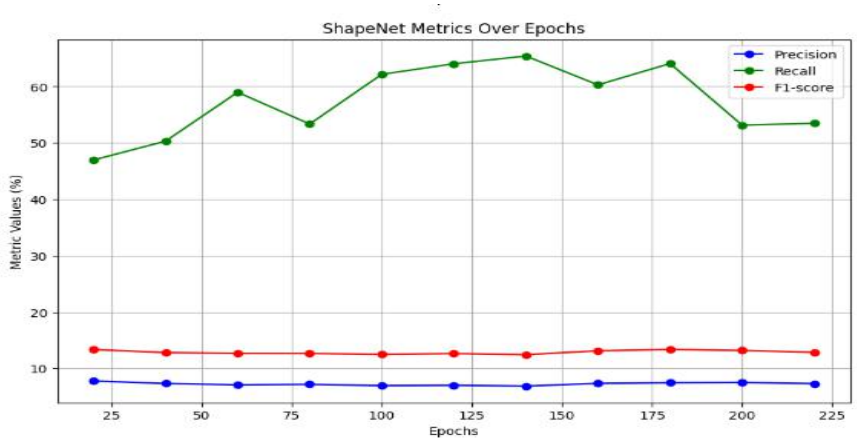


Figure 43- Métriques du modèle CGAN pendant 100 epochs

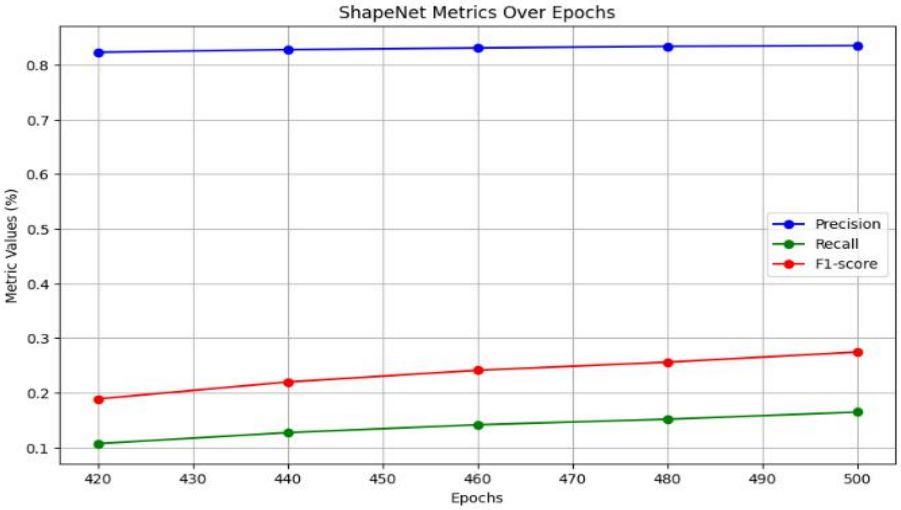


Figure 44- Métriques du modèle auto-encodeur pendant 100 epochs

➢ CGAN seul :

- Rappel raisonnable de 0,6086, indiquant que le modèle détecte environ 60% des formes positives.
- Faible précision de 0,0700, suggérant que seuls 7% des formes générées correspondent réellement à des formes positives, avec un grand nombre de faux positifs.
- F1-score de 0,1254, indiquant un déséquilibre entre la précision et le rappel dû à la faible précision.

- Auto-encodeur :

- ✧ Rappel élevé de 0,8548, indiquant que le modèle détecte environ 85% des formes positives présentes dans les données réelles.
- ✧ Précision élevé de 0,8, suggérant que 80% des formes détectées comme positives sont réellement des formes positives, avec un très peu nombre de faux positifs.
- ✧ F1-score relativement faible de 0,1296, indiquant une performance médiocre en termes d'équilibre entre la précision et le rappel.
- ✧ La fonction de perte de l'auto-encodeur reste toujours en 1200 ce qui peut être améliorer

8. Discussion :

D'après nos études, nous avons pu constater qu'en ajoutant un auto-encodeur pour décoder l'image générée par le CGAN, cela a donné de meilleurs résultats qu'avec le CGAN seul. Pour que le générateur soit plus performant, nous devons trouver un équilibre entre lui et le discriminateur, car c'est à lui de générer les formes à partir des descriptions textuelles. Entraîner davantage le CGAN serait peut-être la clé pour améliorer ses performances.

Quant aux formes générées, le CGAN seul génère un cube, mais en rajoutant un auto-encodeur, on remarque une amélioration dans le résultat généré lors du décodage de ces images. Cela est dû au fait que l'auto-encodeur a été entraîné avec des NRRD à une résolution de 32, et il essaie de reconstruire ces formes. Grâce à cela, nous pouvons voir une évolution dans nos résultats.

9. Conclusion

Dans le dernier chapitre de notre travail nous avons présenté les outils que nous avons utilisés pour mener à bien notre projet. Puis, nous avons évalué le travail que nous avons réalisé tout en discutant des résultats des expériences que nous avons menées.

Conclusion générale

Les dernières années ont vu un intérêt croissant pour l'apprentissage automatique, qui s'est manifesté notamment par l'émergence de nouvelles architectures. Les chercheurs ont ainsi développé des modèles d'apprentissage automatique innovants, tels que les réseaux neuronaux et les modèles génératifs. Ces derniers ont trouvé application dans de nombreux domaines, notamment en traitement d'images et traitement de texte. Cependant, le domaine de la génération de formes 3D a été relativement peu exploré jusqu'à présent. Il est essentiel de poursuivre les recherches dans ce domaine pour développer des méthodes et des technologies innovantes. La génération de formes 3D a des applications multiples, notamment en design assisté par ordinateur, en vision par ordinateur et en robotique.

Pour réaliser notre travail, nous avons exploité les formes 3D de chaises et de tables issues du dataset ShapeNet, accompagnées de leurs descriptions textuelles. On a d'abord converti les descriptions en vecteurs numériques pour les utiliser comme entrées de nos modèles de génération, pour cela on a utilisé le modèle Transformer pré-entraîné 'all-MiniLM-L6-v2'. Ensuite, nous avons implémenté un auto-encodeur dans le but de réduire la dimensionnalité des formes 3D et d'extraire leurs embeddings, permettant ainsi d'accroître les performances. L'objectif principal était d'utiliser trois approches complémentaires : les réseaux antagonistes génératifs conditionnels (CGAN), CGAN avec auto-encodeur et les modèles de diffusion novateurs. Les CGAN établissent les liens entre les formes générées et leurs descriptions textuelles vectorisées. Les auto-encodeurs permettent d'obtenir les embeddings des formes 3D. Quant aux modèles de diffusion, ils offrent une technique innovante pour générer des formes 3D de haute qualité. Enfin, nous avons analysé et discuté les résultats obtenus avec cette architecture intégrant ces trois méthodes de pointe.

Pour des recherches supplémentaires, on voudrait approfondir l'utilisation (CGAN) dans le contexte de la génération de formes 3D, en explorant notamment des architectures plus complexes capables de produire des détails plus fins et des structures plus élaborées. Il serait également intéressant d'étudier comment les CGAN pourraient être adaptés pour générer non seulement la géométrie des objets 3D, mais aussi leurs propriétés de surface, leurs textures et leurs comportements physiques simulés, ouvrant ainsi la voie à des applications plus avancées dans des domaines tels que la réalité virtuelle, le design industriel et la simulation d'environnements 3D réalistes.

Références bibliographiques

[1] Poulat, A., Martin, T. et Giacomo, G. (2022). Modélisation 3D et génération de descriptions textuelles : Un pont entre l'image et le langage. Dans J. Dupont (dir.), Avancées récentes en infographie et traitement du langage naturel (p. 23-47). Presses universitaires de Lyon.

[2] Alpaydin, E. (2020). Introduction au machine learning (3e éd., Traduit par S. Bengio). Éditions Technip.

[3] Burkov, A. (2019). The Hundred-Page Machine Learning Book (pp. 23-30). Éditions Québec Livres.

[4] https://www.ejable.com/tech-corner/ai-machine-learning-and-deep-learning/types-of-machine-learning/

[5] Chollet, F. et Allaire, J.J. (2018). Deep Learning with R. Manning Publications Co., p.89.

[6] https://parlonssciences.ca/ressources-pedagogiques/documents-dinformation/introduction-a-lapprentissage-machine

[7] Chollet, F. et Allaire, J.J. (2018). Deep Learning with R. Manning Publications Co., p.90.

[8] https://techvidvan.com/tutorials/reinforcement-learning/

[9] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539

[10] Chollet, F. et Allaire, J.J. (2018). Deep Learning with R. Manning Publications Co., p. 7.

[11] https://towardsdatascience.com/mit-6-s094-deep-learning-for-self-driving-cars-2018-lecture-1-notes-807be1a50893

[12] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning (pp. 179-197). Cambridge, MA: MIT Press

[13] https://lucidar.me/en/neural-networks/most-popular-activation-functions-for-deep-learning/

[14] Aggarwal, C. C. (2018). Neural Networks and Deep Learning: A Textbook (p. 57). Cham, Suisse: Springer.

[15] Lauzon, F. Q. (2012). An introduction to deep learning. Proceedings of the 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA), 1438-1439. doi: 10.1109/ISSPA.2012.6310529

[16] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[17] Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[18] Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. Neural computation, 29(9), 2352-2449.

[19] Taylor, M. (2017). Neural Network Visual Introduction for Beginners. No Starch Press, p. 15.

[20] Tyagi, A.K. et Abraham, A. (Eds.) . Recurrent Neural Networks: Concepts and Applications. CRC Press, p.11.

[21] Tyagi, A.K. et Abraham, A. (Eds.) . Recurrent Neural Networks: Concepts and Applications. CRC Press, p.12.

[22] https://www.researchgate.net/figure/Architecture-classique-dun-reseau-de-neurones-convolutif-Une-image-est-fournie-en_fig5_330995099

[23] Tyagi, A.K. et Abraham, A. (Eds.) Recurrent Neural Networks: Concepts and Applications. CRC Press, p.12.

[24] Ahmed Menshawy.deep learning by example,p.259.

[25] Bank, D., Koenigstein, N., & Giryes, R. (2021). Autoencoders. arXiv preprint arXiv:2003.05991.

[26] Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2e éd., pp. 523-525). O'Reilly Media.

[27] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press. (Chapitre 14: Autoencoders)

[28] https://fr.linkedin.com/pulse/ia-les-encodeurs-philippe-jean-baptiste-executive-mba-mpa-msc-ma-m9q0e

[29] Goodfellow et al., 2014, "Generative Adversarial Nets".

[30] Raff, E. (2019). Inside Deep Learning: Math, Algorithms, Models (K. Borne, foreword, p. vii-xi). Pearson, p. 348.

[31] Raff, E. (2019). Inside Deep Learning: Math, Algorithms, Models (K. Borne, foreword, p. vii-xi). Pearson, p. 380.

[32] Rimsha Goomerb, Ashutosh Kumar Singh Jitendra Kumara, "Long Short Term Memory Recurrent Neural Network (LSTM-RNN)," in 6th International Conference on Smart Computing and Communications, kurkushetra india, 2017

[33] https://medium.com/@mayank.bali/sign-language-detection-for-deaf-using-deep-learning-mediapipe-u-opencv-4c5151e2374c

[34] https://www.ibm.com/fr-fr/topics/natural-language-processing

[35] https://www.unite.ai/fr/mod%C3%A8les-de-diffusion-en-ai-tout-ce-que-vous-devez-savoir/

[36] "3D Modeling Definition". TechTarget, 25 mars 2015, https://whatis.techtarget.com/definition/3D-modeling. Accédé le 13 mars 2024.

[37] Nguyen, T., Asmaa, E. & Ardabilian, M. (2021). 3D Imaging, Analysis and Applications. Springer, pp. 267.

[38] Nguyen, T., Asmaa, E. & Ardabilian, M. (2021). 3D Imaging, Analysis and Applications. Springer, pp. 268.

[39] Remeshing Techniques for Interactive Geometry Processing" par Michael Botsch et Leif Kobbelt (2004)

[40] https://c.mql5.com/2/45/mesh.png

[41] Nguyen, T., Asmaa, E. & Ardabilian, M. (2021). 3D Imaging, Analysis and Applications. Springer, pp. 6, 38, 280.

[42] Darian, J. (2018, Août 30). Comparaison entre représentations pixel 2D et voxel 3D [Image]. Medium. https://medium.com/@julie_93447/3d-learning-its-time-to-teach-in-voxels-instead-of-pixels-d644df659328

[43] Curless, B., Levoy, M., Cohen, D., Sheffer, A., & Ericson, C. (1994-2004). Voxel Representation for Computer Graphics, Collision Detection, and Geometric Processing.

[44] Section 3.1 de l'article "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

[45] Reimers, N. et Gurevych, I. (2019). Architecture du modèle Siamois SBERT [Figure]. Repéré sur https://www.researchgate.net/figure/SBERT-Siamese-model-architecture_fig3_378440592

[46] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv preprint arXiv:1908.10084.

[47] Chen, K., Choy, C. B., Savva, M., Chang, A. X., Funkhouser, T., & Savarese, S. (2018). Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings. arXiv:1803.08495v1 [cs.CV] 22 Mar 2018, pp. 1-35

[48] Lee, H., Savva, M., & Chang, A.X. (2022). Text-to-3D Shape Generation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

[49] Shriram, J., Trevithick, A., Liu, L., & Ramamoorthi, R. (2023). RealmDreamer: Text-Driven 3D Scene Generation with Inpainting and Depth Diffusion. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

[50] Yang, X., & Song, P. (2022). Adversarial Synthesis of 3D Object Shape and Appearance from Sketches and Text. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[51] https://www.sbert.net/docs/pretrained_models.html

[52] Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

[53] Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

[54] Jiang, L., Cai, J., Zheng, J., Xu, H., Yuan, J., Zhou, K., ... & Bao, H. (2021). Conditional GAN for 3D Controllable Portrait Generation. ACM Transactions on Graphics (TOG), 40(4), 1-14.

[55] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of stylegan. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8110-8119).

[56] Yang, J., Kannan, A., Batra, D., & Parikh, D. (2017). LR-GAN: Layered recursive generative adversarial networks for image generation. arXiv preprint arXiv:1703.01560.

[57] cloud.google.com/colaboratory /gpu

[58] https://www.python.org

[59] https://numpy.org/

[60] Kindlmann, G. (2003). Semi-automatic generation of transfer functions for direct volume rendering (Thèse de doctorat). Université Cornell.

[61] Devlin, J., Chang, M.-T., Lee, K. and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL].

[62] https://www.datarockstars.ai/pytorch-la-bibliotheque-de-deep-learning-de-meta/

[63] Kaggle,Available at : http://www.kaggle.com.

[64] https://www.geeksforgeeks.org/introduction-to-tkinter/

[65] https://visualstudio.microsoft.com/fr/

[66] https://matplotlib.org/

[67] https://shapenet.org/

[68] Chen, K., Choy, C. B., Savva, M., Chang, A. X., Funkhouser, T., & Savarese, S. (2018). Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings. arXiv:1803.08495v1 [cs.CV] 22 Mar 2018, pp. 1-35

[69] A. Géron, (2019) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (p.120). O'Reilly Media.

[70] B. Mohamed, Deep Learning for Detecting and Identifying Blinding Retinal Diseases, Thèse de master LMD en Informatique, sous la direction de Abdelouahab, page 48.