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 : **Ingénierie des logiciels**

THÈME

Approche de Génération Des Formes 3D À Partir Des Descriptions Textuelles en LLM

Présenté Par :

ABDICHE Imene & DERDER Narimane

Soutenu Publiquement le : /06/2023

Devant le jury composé de :

Encadreur: Mr A.H KAMECHE **Président (e) :** Mme S.Oukid

Examinateur : Mme K.MIDOUN

Année Universitaire : 2022/2023

Résumé:

Au fil des années, l'utilisation croissante de modèles 3D dans diverses industries, telles que la réalité virtuelle et l'architecture, a suscité un intérêt croissant pour la génération de formes 3D. Cependant, la conception et la création de formes 3D précises et réalistes peuvent être coûteuses en termes de temps et de ressources.

Dans notre projet, nous avons cherché à minimiser ces coûts et à simplifier le processus en explorant la génération de formes 3D à partir de descriptions textuelles. Pour ce faire, nous avons utilisé une variante des réseaux de neurones antagonistes génératifs (CGAN), ainsi que le modèle d'Auto-Encodeur et un modèle Transformer pré-entraîné pour traiter les entrées textuelles. Nos modèles ont été entraînés et évalués en utilisant le jeu de données "ShapeNet", qui contient des informations sur des objets tels que des tables et des chaises.

Mots clé: Génération de formes 3D, descriptions textuelles, CGAN, Auto-Encoder, Transformer, ShapeNet.

Abstract:

Over the years, there has been a growing interest in 3D shape generation due to the increasing use of 3D models in industries like virtual reality and architecture. However, creating precise and realistic 3D shapes can be time-consuming and resource-intensive. In our project, we aimed to address this issue by exploring the generation of 3D shapes from textual descriptions, with the goal of minimizing costs and simplifying the process.

To achieve this, we employed a variant of generative adversarial networks (GANs) called Conditional GANs (CGAN). In addition, we utilized the Auto-Encoder model and a pre-trained Transformer model to process textual inputs. These models were trained and evaluated using the "ShapeNet" dataset, focusing specifically on tables and chairs.

Key words: Generation of 3D shapes, textual descriptions, CGAN, Auto-Encoder, Transformer, ShapeNet.

ملخص: على مر السنين، زاد استخدام النماذج ثلاثية الأبعاد في مختلف الصناعات، مثل الواقع الافتراضي والهندسة المعمارية، مما أثار اهتمامًا متزايدًا في توليد الأشكال ثلاثية الأبعاد. ومع ذلك، يمكن أن يكون تصميم وإنشاء أشكال ثلاثية الأبعاد دقيقة وواقعية مكلفًا من حيث الوقت والموارد.

في مشروعنا، سعينا إلى تقليل هذه التكاليف وتبسيط العملية عن طريق استكشاف توليد الأشكال ثلاثية الأبعاد من الوصف النصي. للقيام بذلك، استخدمنا إصدارًا معدلًا *من الشبكات العصبية الإنشائية* **(CGAN)** ، بالإضافة إلى نموذج **Auto-Encoder** ونموذج **Transformer** مدرب مسبقًا لمعالجة الإدخالات النصية. تم تدريب وتقييم نماذجنا باستخدام مجموعة البيانات **«ShapeNet»**، التي تحتوي على معلومات حول أشياء مثل الطاولات والكراسي.

الكلمات المفتاحية: توليد أشكال ثلاثية الأبعاد, وصفات نصية, CGAN**,** Auto-Encoder, Transformer, ShapeNet.

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.

Narimane et Imene

Dédicace

Je souhaite dédier ce travail à ma famille qui a été ma source de soutien inconditionnel. Je tiens à exprimer ma gratitude envers mes parents, mes frères Walid et Imad, ainsi que mes sœurs Sarah, Ania et Dania. Votre amour, vos encouragements et votre confiance ont été mes principales motivations tout au long de mes études. Votre fierté et votre soutien indéfectible m'ont donné la détermination nécessaire pour atteindre mes objectifs.

Grand-père, même si tu n'es plus parmi nous, ton amour et ta présence éternelle continuent de briller intensément dans nos cœurs. Tu as laissé une empreinte inaltérable dans ma mémoire et je suis honorée de te dédier ce travail.

Je tiens également à exprimer ma sincère gratitude envers mon binôme, Narimane, 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.

À mon fiancé, H. Moussa, je veux te dire merci pour ton amour, ton soutien inconditionnel et ta compréhension pendant cette période exigeante. Tu as été ma source d'inspiration et de réconfort, et j'ai la chance de t'avoir à mes côtés.

Je tiens à remercier tous mes amis, en particulier Khadîdja, Chaïma, Yusra et Fouzia, 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

Imene

Dédicace

Je dédie ce travail à tous mes proches.

Narimane

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

Abbreviation	Description complete
GAN	Generative Adversarial Networks
CGAN	Conditional Generative Adversial Network
CNN	Convolutional Neural Network
MSELoss	Mean Squared Error Loss
ReLU	Rectified Linear Unit

Introduction Générale

Au fil des années, la demande de la conception et la création d'objets en trois dimensions a augmentée, grâce aux avancées technologiques. Aujourd'hui, de nombreuses industries telles que l'architecture, le cinéma, les jeux vidéo, la médecine ou l'industrie automobile, ont besoin de modèles 3D précis et réalistes. Cependant, la création de ces modèles peut prendre beaucoup de temps et de ressources. C'est pourquoi la génération de formes 3D à partir de descriptions textuelles suscite un grand intérêt dans le domaine de l'intelligence artificielle.

Problématique

La création de modèles 3D précis et réalistes peut être une tâche coûteuse en termes de temps et de ressources. L'approche traditionnelle de conception de formes 3D nécessite une expertise et une intervention manuelle importantes. La génération de formes 3D à partir de descriptions textuelles représente une solution potentielle à ce problème. Cependant, la capacité de générer des formes 3D à partir de descriptions textuelles qui ne font pas partie du jeu de données d'entraînement pose un défi supplémentaire.

Notre étude se confronte donc à la problématique suivante : Comment développer un modèle capable de générer des formes 3D précises et réalistes de tables et de chaises à partir de descriptions textuelles, tout en prenant en compte la généralisation à de nouvelles descriptions qui n'ont pas été observées lors de l'entraînement ? Nous devons trouver des solutions pour apprendre les relations entre le texte et les formes 3D, tout en garantissant la flexibilité du modèle pour générer des formes correspondant à une grande variété de descriptions textuelles.

Objectifs du travail

L'objectif de notre étude est de développer un modèle qui peut générer des formes 3D spécifiques de tables et de chaises à partir de n’importe quelles descriptions textuelles. Pour cela, nous allons proposer une approche basée sur le deep learning, en utilisant spécifiquement des architectures génératives, pour entraîner notre modèle à apprendre les relations entre les mots présents dans les descriptions textuelles et les formes 3D correspondantes.

Organisation du mémoire

Ce mémoire est structuré de la manière suivante :

Chapitre 1 : Nous présentons quelques notions de base de l'apprentissage automatique et de l'apprentissage profond, en mettant l'accent sur le traitement du texte et des formes 3D dans ces domaines. Nous examinons également quelques exemples de travaux connexes liés à l'approche "text to shape".

Chapitre 2 : Nous exposons la structure de notre travail ainsi que les méthodes que nous avons employées pour le réaliser.

Chapitre 3 : Nous présentons les outils et le jeu de données utilisés, ainsi que les expérimentations que nous avons menées et les résultats obtenus.

Chapitre 1 : Apprentissage automatique

1. Introduction :

L'utilisation de l'intelligence artificielle pour générer des formes 3D à partir de textes a connu une croissance rapide ces dernières années. Cette approche, appelée "text to shape", utilise des techniques d'apprentissage automatique et d'apprentissage profond pour créer des modèles qui peuvent comprendre et interpréter des descriptions textuelles et les transformer en objets 3D. Dans ce chapitre, nous allons explorer les bases de l'apprentissage automatique et de l'apprentissage profond appliquées au traitement de texte et de formes 3D, et nous allons examiner quelques exemples de travaux récents en rapport avec l'approche "text to shape".

2. Notions de base :

2.1. L'apprentissage automatique (ou Machine Learning en anglais**) :** Est un domaine de l'intelligence artificielle qui permet aux machines d'apprendre à partir des données et d'améliorer leurs performances sur des tâches spécifiques sans être explicitement programmées [1].

Il existe plusieurs types d'apprentissage automatique. Voici les plus utilisés :

- Supervisé
- Non supervisé
- Apprentissage par renforcement

2.1.1. L'apprentissage supervisé : Consiste à prédire une sortie à partir d'une entrée. Le modèle est entraîné sur un ensemble de données étiquetées, puis utilise ces exemples pour généraliser et prédire des sorties pour de nouvelles entrées non étiquetées [2].

Avec ce type d'apprentissage on peut développer des modèles pour résoudre deux typesde problèmes :

- **Régression :** Cherche à prédire la valeur d'une variable continue à partir d'un ensemble de variables d'entrée. Elle consiste à modéliser la relation entre les variables d'entrée et la variable de sortie à l'aide d'une fonction mathématique [3].

Un exemple de régression est la prédiction du prix d'un appartement en fonction de ses caractéristiques (telles que la superficie, le nombre de chambres, etc..). Les données d'apprentissage peuvent être un ensemble d'appartements vendues précédemment, avec leurs caractéristiques et leur prix de vente. En peut développer un modèle de régression et l'entrainé à prédire le prix d'un nouvel appartement en fonction de ses caractéristiques. La figure 1 représente un exemple de dataset d'entrainement dans ce concept.

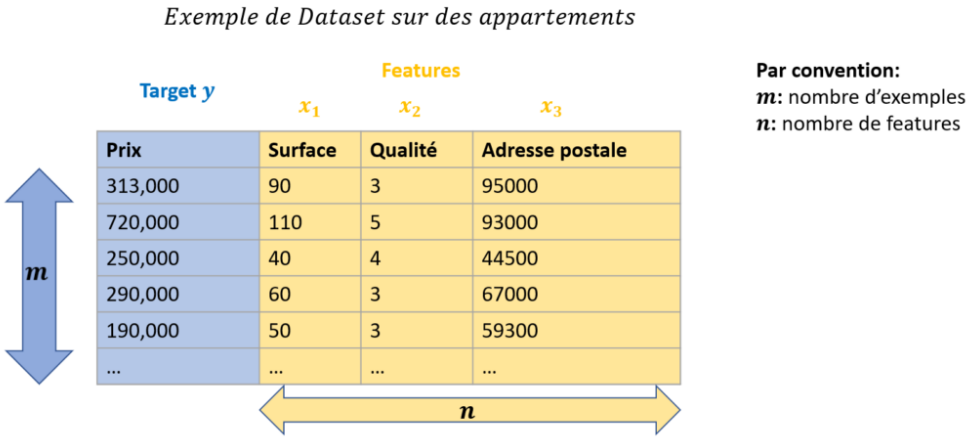

Figure 1 : Exemple de régression, apprentissage supervisé [4]

- **Classification :** Elle consiste à prédire une variable de classe discrète à partir d'un ensemble de caractéristiques ou de variables explicatives. Elle est utilisée pour résoudre des problèmes de catégorisation tels que la reconnaissance d'images [3].

La figure 2 montre un exemple de classification de chaises et de tables :

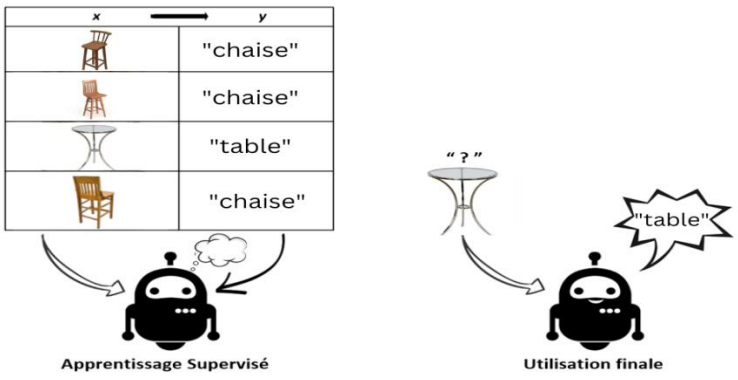

Figure 2: Exemple de classification, apprentissage supervisé [4]

2.1.2. L'apprentissage non supervisé : Est une technique d'apprentissage automatique où un modèle est entrainé sur un ensemble de données non étiquetées. Le but est d'identifier des structures, des modèles ou des relations cachées dans les données [3].

Il existe deux types d'apprentissage non supervisé :

- **Le regroupement (Clustering** en anglais**) :** Cette technique vise à regrouper des données en ensembles homogènes en fonction de leur similarité. Il existe plusieurs méthodes de clustering, notamment la méthode des K-means.
- **L'association de règles (Association Rule Learning** en anglais**) :** Cette technique vise à découvrir des relations entre des éléments d'un ensemble de données. Elle est souvent utilisée dans des domaines tels que la recommandation de produits. Il existe plusieurs méthodes d'association, notamment la méthode Apriori.

La figure 3 représente un exemple pour les deux types :

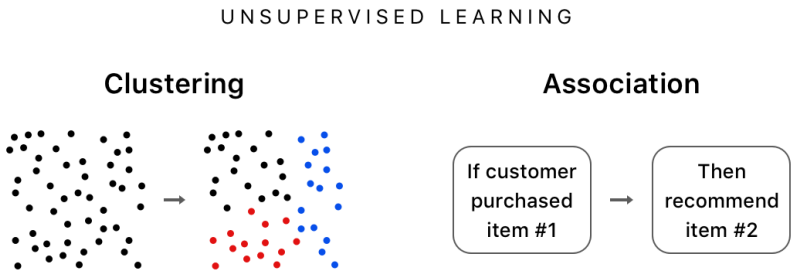

Figure 3 : Apprentissage non supervisé [5]

2.1.3. L'apprentissage par renforcement : Est une méthode d'apprentissage automatique où un agent apprend à travers l'interaction avec un environnement. L'agent prend des décisions en fonction de son état actuel et reçoit des récompenses ou des pénalités en fonction de la qualité de ses actions. L'objectif de l'agent est de trouver la politique (séquence d'actions) qui maximise la récompense totale qu'il reçoit de l'environnement. La figure 4 représente un exemple d'utilisation de l'apprentissage par renforcement dans le jeu Mario [6].

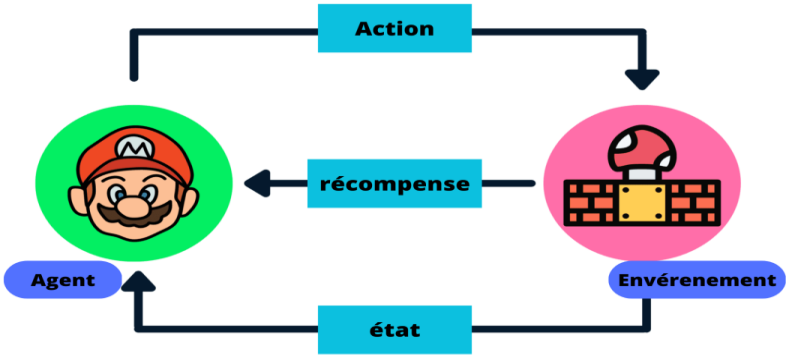

Figure 4 : Exemple d'apprentissage par renforcement [7]

2.2. L'apprentissage profond (**ou Deep Learning** en anglais) : Est une approche de l'apprentissage automatique qui s'inspire du fonctionnement du cerveau humain pour créer des modèles d'apprentissage profonds, constitués de plusieurs couches de neurones artificiels. Ces modèles sont capables de traiter des données de grande dimensionnalité et d'extraire des caractéristiques de plus haut niveau de complexité pour réaliser des tâches telles que la reconnaissance d'images, la traduction automatique, etc. [8]

2.2.1. Les réseaux de neurones artificiels : Sont des modèles de machine learning composés de nombreux neurones interconnectés qui sont organisés en couches. Chaque couche transforme les données d'entrée en une représentation plus abstraite qui est transmise à la couche suivante. Les réseaux de neurones sont capables d'apprendre des motifs complexes à partir des données en ajustant les poids de connexion entre les neurones à l'aide d'algorithmes d'optimisation [10].

2.2.2. Architecture globale des réseaux de neurones : Les réseaux de neurones sont généralement constitués de plusieurs couches, qui comprennent des neurones interconnectés et des poids associés à ces connexions. On distingue généralement trois types de couches : les couches d'entrée, les couches cachées et les couches de sortie. Les données sont introduites dans la couche d'entrée. Les informations sont ensuite propagées à travers les couches cachées, qui effectuent des transformations non linéaires sur les données. La dernière couche correspond à la couche de sortie, qui produit les prédictions du modèle. Les poids de connexion entre les neurones sont ajustés lors de l'apprentissage à l'aide d'algorithmes d'optimisation. La figure 6 illustre d'une manière simple cette architecture [9].

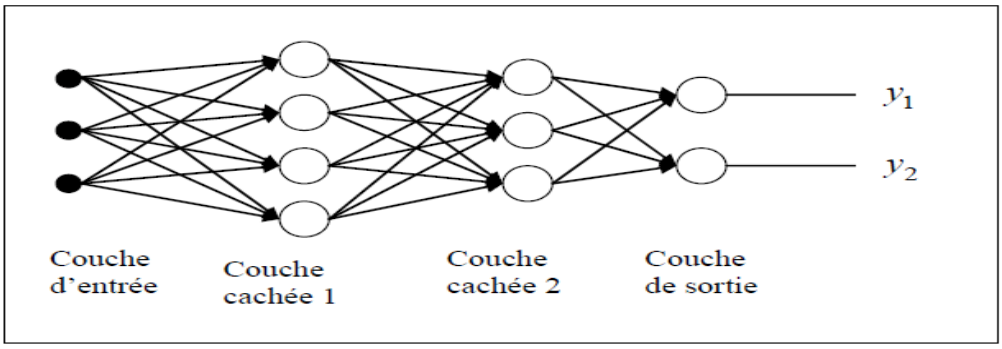

Figure 5 : Architecture globale des réseaux de neurones [11]

2.3. Fonction d'activation : Est une fonction mathématique appliquée à la sortie d'un neurone dans un réseau de neurones. Elle est utilisée pour introduire de la non-linéarité dans le modèle et réguler la sortie du neurone. Les fonctions d'activation les plus courantes sont : la fonction sigmoïde, la fonction tangente hyperbolique et la fonction 'ReLU' [9].

2.4. Fonction de perte (ou de coût) : Mesure l'écart entre la sortie prédite d'un modèle de machine learning et la sortie réelle attendue [9].

2.5. Algorithmes d'optimisation : Sont utilisés pour ajuster les paramètres d'un modèle de machine learning afin de minimiser une fonction de coût. Les algorithmes les plus couramment utilisés sont : la descente de gradient stochastique (SGD), l'ADAM et l'AdaGrad. Ils diffèrent dans leur manière de mettre à jour les poids du modèle [9].

3. Exemples d'architectures de réseaux de neurone (ou d'algorithmes de Deep Learning) : Il existe une multitude d'architectures de réseaux de neurones, voici quelques unes :

3.1. Les réseaux de neurones convolutionnels (CNN) : Sont une classe de réseaux de neurones de base, largement utilisée pour le traitement des images et des vidéos.

L'architecture des CNN consiste en plusieurs couches, notamment des couches de convolution, des couches de pooling et des couches entièrement connectées [12].

La couche de convolution : C'est la couche la plus importante des CNN. Elle applique un filtre sur les pixels de l'image d'entrée pour extraire les caractéristiques importantes. Le filtre se déplace sur l'image d'entrée en effectuant une opération de multiplication et d'addition pour calculer une valeur unique appelée activation. Cette opération est répétée pour tous les pixels de l'image d'entrée, produisant une carte d'activation. La formule mathématique de la convolution est la suivante :

$s(t) = \int x(a)\, w\, (t - a)\, da$, où x est l'image d'entrée, w est le noyau de convolution (également appelé

filtre) et s est la sortie [12].

La figure 7 représente un exemple d'application d'un filtre sur une partie d'une image d'entré, avec les calcules nécessaires.

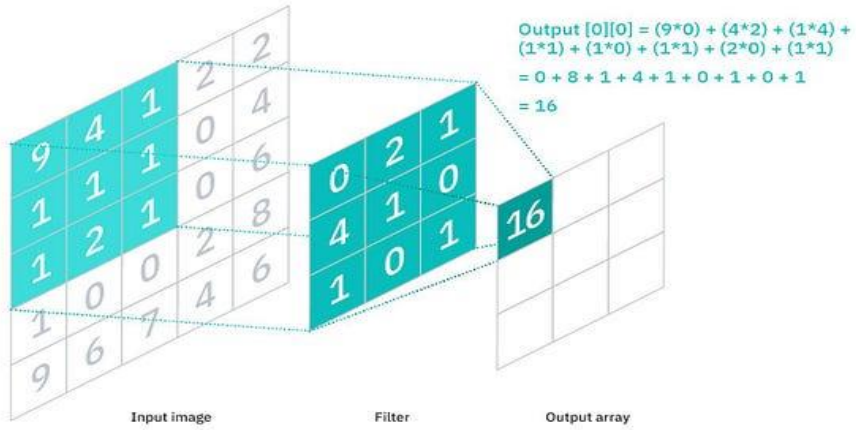

Figure 6 : Exemple de l'application d'un filtre, couche de convolution, CNN [13]

La couche de pooling : Utilisée pour réduire la taille de l'image en effectuant une opération de sous-échantillonnage. Cette opération permet de réduire le nombre de paramètres du modèle, ce qui peut aider à prévenir le sur-apprentissage. Les opérations de pooling courantes sont la moyenne, le maximum et la médiane. La figure 8 représente un exemple de l'application de l'opération 'max pooling' sur une carte de caractéristiques [12].

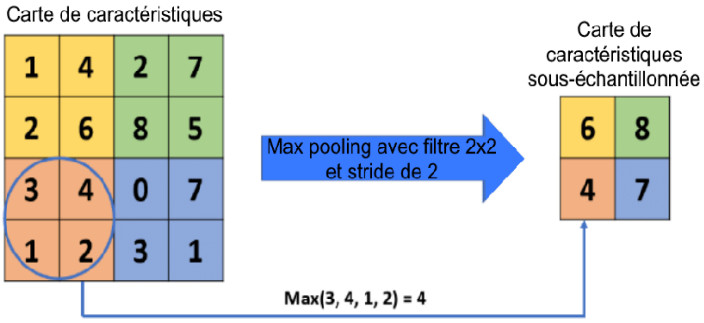

Figure 7 : Exemple de Max pooling, couche de pooling, CNN [14]

Couche entièrement connectée : Cette couche est similaire à celle des réseaux de neurones classiques. Elle est responsable de la classification ou de la prédiction finale en transformant les caractéristiques extraites de l'image d'entré en une représentation adaptée à la tâche spécifique. La figure 9 représente un exemple de classification d'image par un modèle CNN [8].

Les CNN sont utilisés dans de nombreux domaines, notamment la détection de visages, la

classification de texte et l'analyse de sentiments.

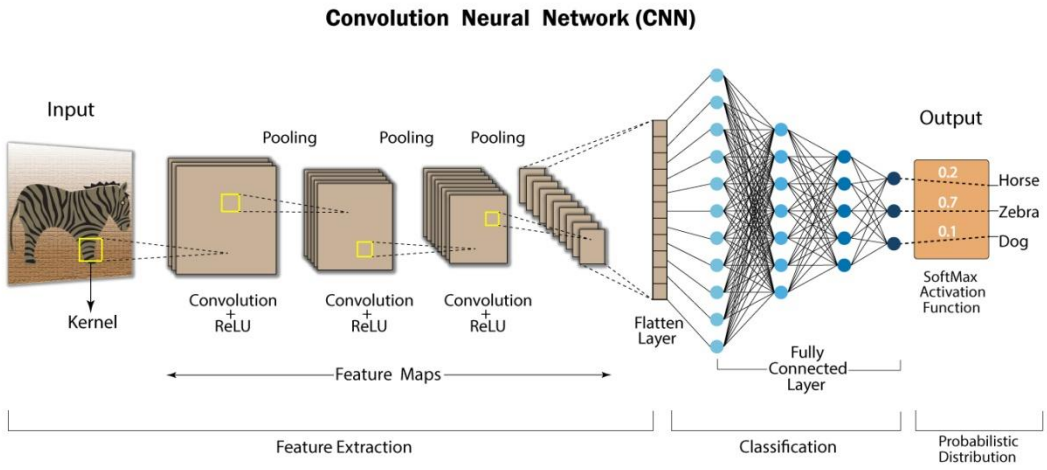

Figure 8 : Exemple qui illustre l'architecture des CNN [15]

3.2. Les réseaux adversaires génératifs (GAN) et les réseaux adversaires conditionnels (CGAN): Sont deux architectures de réseaux de neurones artificiels utilisées pour générer des données synthétiques à partir de données d'entrée. Le CGAN est une extension de GAN classique, leurs architectures sont basées sur un cadre de modélisation générative.

Dans ces deux architectures, deux réseaux neuronaux sont en compétition : un générateur et un discriminateur.

Dans un GAN, le générateur prend des vecteurs de bruit aléatoires en entré et génère des échantillons en sortie, tels que des images, dans le but de tromper le discriminateur. Le discriminateur, quant à lui, essaie de distinguer les échantillons générés par le générateur des vrais échantillons provenant d'un ensemble de données réel (figure 10).

L'idée centrale des GAN est que le générateur et le discriminateur s'améliorent mutuellement lors d'un processus d'entraînement itératif. Le générateur apprend à générer des échantillons de plus en plus réalistes pour tromper le discriminateur, tandis que le discriminateur apprend à mieux distinguer les vrais échantillons des faux. Ce processus d'entraînement en compétition permet au générateur de capturer les caractéristiques essentielles des données réelles [16].

Dans le cas des CGAN, un aspect supplémentaire est introduit : la condition. Les CGAN utilisent des informations conditionnelles supplémentaires en entré pour guider le processus de génération. Par exemple, dans la génération d'images, les informations conditionnelles peuvent être des étiquettes de classe spécifiant le type d'objet à générer. Cela permet de contrôler de manière plus précise la génération des échantillons (figure 11).

Voici comment fonctionnent les CGANs avec un exemple de fonctions de cout couramment utilisées dans les GAN :

Soit X une donnée d'entrée et Y une condition. Le but des CGANs est de générer une donnée synthétique Y' à partir de Y, tout en garantissant que Y' soit aussi proche que possible de X.

Le générateur G prend en entrée un bruit aléatoire Z et une condition Y, et produit une donnée synthétique Y' : Y' = G(Z, Y)

Le discriminateur D prend en entrée soit X et Y, soit Y' et Y, et prédit si la donnée est vraie (X) ou fausse (Y'). La fonction de coût du discriminateur est définie comme suit :

- Pour les vraies données X : L_D_real = -log(D(X, Y))
- Pour les données synthétiques Y' : L_D_fake = -log(1 - D(Y', Y))
- La fonction de coût totale du discriminateur est L_D = L_D_real + L_D_fake

Le générateur G est entraîné pour minimiser la fonction de coût du discriminateur, tout en maximisant la probabilité que le discriminateur considère les données synthétiques comme vraies. La fonction de coût du générateur est définie comme suit :

- L_G = -log(D(Y', Y))

Le but de l'apprentissage des CGANs est de minimiser la fonction de coût du générateur L_G tout en maximisant la fonction de coût du discriminateur L_D [17].

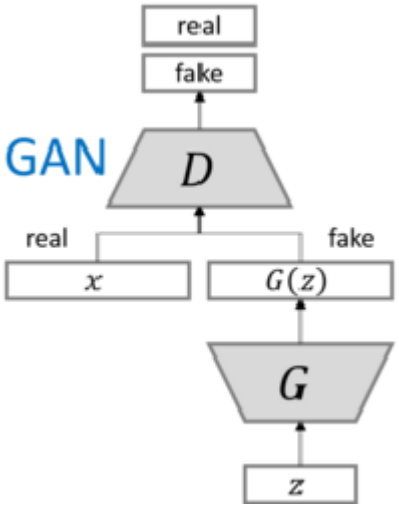

Figure 9 : Architecture de GAN [18]

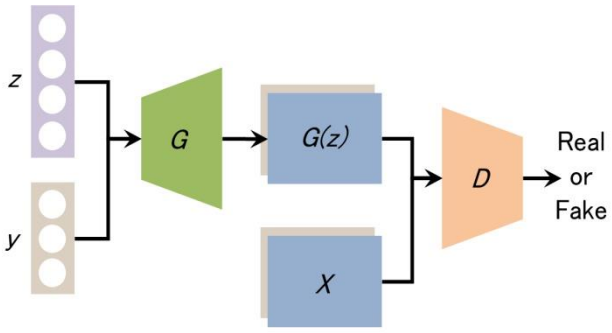

Figure 10 : Architecture du CGAN [19]

3.3. Les Autoencodeurs : Sont des réseaux de neurones artificiels qui apprennent à représenter les données d'entrée dans un espace de dimension réduite, appelé espace latent, en appliquant une fonction d'encodage. Ils sont ensuite capables de reconstruire les données d'entrée initiales à partir de l'espace latent en appliquant une fonction de décodage. Les autoencodeurs sont souvent utilisés pour

la compression de données et la génération de données.

L’Espace latent : Est une représentation compressée des données d'entrée obtenue à partir de l'encodeur. Il contient une représentation plus concise des caractéristiques importantes des données d'entrée.

Voici les équations mathématiques qui décrivent le fonctionnement des autoencodeurs :

Soit X un vecteur de données d'entrée de dimension n, et Y le vecteur de sortie de dimension n', où n' peut être égal à n ou être une valeur inférieure (figure 12).

1. Encodage : h = f(Wx + b) où W est la matrice des poids des connexions entre les neurones d'entrée et les neurones de la couche cachée (espace latent), b est le vecteur de biais de la couche cachée et f est la fonction d'activation. Le vecteur h est le résultat de l'encodage de X dans l'espace latent.
2. Décodage : Y = g(Vh + c) où V est la matrice des poids des connexions entre les neurones de la couche cachée et les neurones de sortie, c est le vecteur de biais de la couche de sortie et g est la fonction d'activation. Le vecteur y est le résultat de la reconstruction de X à partir de l'espace latent.
3. Fonction de perte : L(X, Y) = ||X - Y||^2 où ||.|| est la norme Euclidienne. La fonction de perte mesure la différence entre les données d'entrée initiales X et les données reconstruites Y.

Le but de l'apprentissage des autoencodeurs est de minimiser la fonction de perte L(X, Y) en ajustant les poids et les biais des connexions entre les neurones d'entrée, la couche cachée et les neurones de sortie.

Les autoencodeurs sont utilisés dans de nombreuses applications, telles que la réduction de la dimensionnalité des données, la génération de données, la détection d'anomalies, la segmentation d'images, etc. Ils ont été appliqués avec succès dans des domaines tels que la vision par ordinateur, la reconnaissance de la parole, le traitement du langage naturel, etc. [20]

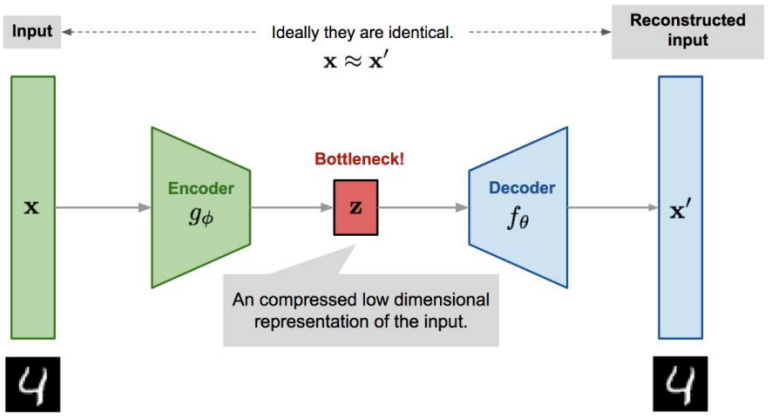

Figure 11 : Exemple qui illustre l'architecture des auto-encodeurs [21]

3.4. Les Transformers : Sont une architecture de réseau de neurones pour le traitement de séquences, tels que des phrases ou des séquences de symboles.

L'architecture des transformers est composée de deux parties principales : les encodeurs et les décodeurs. Chacune de ces parties contient un certain nombre de couches, et chaque couche contient plusieurs modules d'attention multi-têtes. Les couches des encodeurs et des décodeurs ont des rôles différents, mais elles sont toutes construites sur le même principe de l'attention multi-têtes (figure 13).

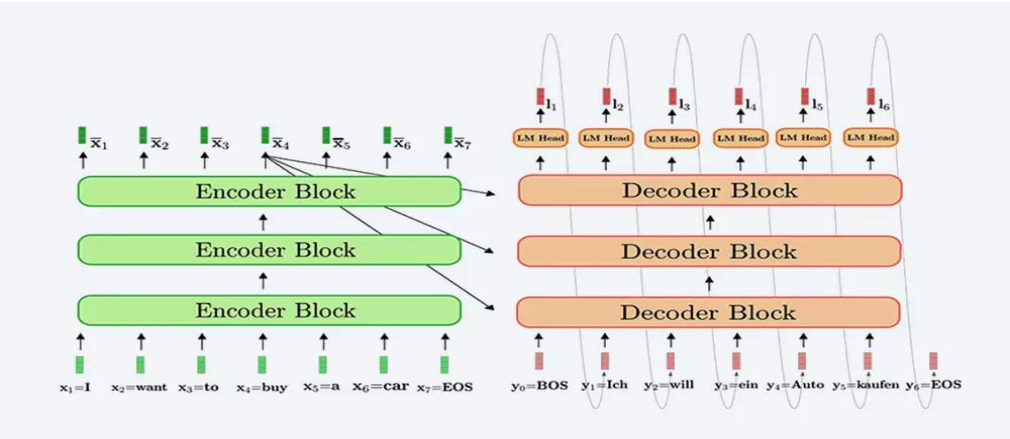

Figure 12 : Architecture des Transformers [22]

Un encodeur prend une séquence d'entrée et la transforme en une représentation vectorielle. Chaque couche d'encodeur contient deux modules : un module d'attention multi-têtes et un module d'alimentation avant (feedforward). Le module d'attention multi-têtes permet au réseau de se concentrer sur différentes parties de la séquence en même temps, tandis que le module d'alimentation avant utilise une couche de neurones entièrement connectée pour transformer la représentation de la séquence.

Un décodeur prend une représentation vectorielle de la séquence d'entrée générée par les encodeurs et génère une séquence de sortie. Chaque couche de décodeur contient également deux modules : un module d'attention multi-têtes et un module d'alimentation avant. En plus de ces modules, le décodeur utilise également un module de masquage pour empêcher le modèle d'accéder aux informations futures de la séquence de sortie [23].

4. Traitement de texte et modèles de représentation de texte :

Consiste à développer des modèles d'apprentissage automatique pour analyser, comprendre et traiter les données textuelles de manière automatisée. Pour cela, le texte est généralement transformé en vecteurs numériques qui servent d'entrée aux modèles. Il existe plusieurs techniques pour transformer le texte en vecteur, telles que : les sacs de mots, les embeddings de mots, les modèles de langage, etc. Ces techniques permettent de représenter le texte sous forme de données numériques compréhensibles par les modèles d'apprentissage automatique [24].

Voici quelques modèles de langage pour le traitement de texte en machine Learning :

- **GloVe (Global Vectors for Word Representation) :** C'est un modèle d'apprentissage non supervisée utilisée pour représenter les mots sous forme de vecteurs numériques (embeddings). Ces embeddings capturent les relations sémantiques et syntaxiques entre les mots en analysant les co-occurrences statistiques des mots dans un corpus de texte [25].
- **CBOW (Continuous Bag-of-Words) :** C'est un modèle d'apprentissage automatique utilisé pour générer des embeddings de mots. Il s'agit d'une méthode de prédiction de mots basée sur le contexte. Le modèle CBOW tente de prédire un mot donné en utilisant le contexte des mots environnants dans une fenêtre fixe [26].
- **Skip-gram :** C'est un modèle d'apprentissage automatique utilisé pour générer des embeddings de mots. Il fonctionne de manière inverse à CBOW. Il utilise un mot donné pour prédire les mots environnants dans un contexte. Il est souvent utilisé pour capturer les relations sémantiques entre les mots et est plus adapté aux corpus de texte volumineux [26].
- **LLM (Language Model) :** Un LLM est un modèle statistique ou un modèle d'apprentissage automatique qui est capable de générer du texte ou de prédire la probabilité d'une séquence de mots dans un langage donné. Il est généralement formé sur de grandes quantités de données textuelles pour apprendre les relations et les structures linguistiques, et peut être utilisé pour des tâches telles que la traduction automatique, la génération de texte et la correction automatique [27].

5. Forme 3D :

Se réfèrent aux objets et modèles tridimensionnels qui ont une profondeur et des dimensions dans les trois directions de l'espace [28].

Dans le domaine du machine learning, les formes 3D peuvent être utilisées comme données d'entrée pour entraîner des modèles d'apprentissage automatique. Les formes 3D peuvent être prétraitées et transformées en formats compatibles avec les modèles d'apprentissage automatique. Voici quelques méthodes couramment utilisées pour la représentation des formes 3D en machine learning :

- **Les voxels (volumetric pixels) :** Sont des éléments tridimensionnels équivalents aux pixels dans les images 2D. Ils représentent des points discrets dans un espace volumétrique tridimensionnel. Chaque voxel contient des informations sur la présence ou l'absence d'une partie de l'objet à une position spécifique dans l'espace. La représentation en voxels est souvent utilisée pour modéliser des objets volumétriques tels que des organes dans l'imagerie médicale ou des objets 3D dans la reconstruction [29].
- **Points en nuage (Point Clouds) :** Les points en nuage sont des ensembles non structurés de points tridimensionnels représentant une forme 3D. Chaque point du nuage de points contient des informations sur les coordonnées spatiales et peut également être accompagné de caractéristiques supplémentaires telles que des couleurs ou des intensités. Cette représentation capture la géométrie de la surface d'un objet 3D sans spécifier explicitement la connectivité entre les points. Les points en nuage sont souvent utilisés dans des tâches telles que la reconnaissance d'objets, la reconstruction de scènes et la modélisation géométrique [30].
- **Maillage (Mesh) :** Un maillage est une représentation composée de faces, d'arêtes et de sommets qui décrivent la géométrie d'un objet 3D. Les faces sont des surfaces polygonales qui définissent les parties de l'objet, les arêtes sont les lignes entre les faces, et les sommets sont les points de connexion des arêtes. Cette représentation permet de capturer la topologie et la connectivité des surfaces et est couramment utilisée dans des tâches telles que la modélisation 3D, la simulation physique, la création d'animations et la reconstruction à partir de données volumétriques [23].

Le choix de la méthode de représentation dépend de la tâche spécifique et des caractéristiques des formes 3D à traiter. Chaque méthode a ses avantages et ses limitations en termes de complexité de calcul, de précision et de représentation des détails géométriques.

Les formes 3D en machine learning sont utilisées dans divers domaines, tels que la vision par ordinateur, la robotique, la réalité virtuelle, la modélisation 3D, etc. [31]

6. Travaux connexes : On peut distinguer deux types d'approches pour la génération de formes 3D à partir de descriptions textuelles :

6.1. La génération directe de formes 3D à partir de texte :

Cette approche consiste à générer directement une forme 3D à partir d'une description textuelle. Elle nécessite une compréhension profonde des données textuelles pour extraire les informations pertinentes et les convertir en une représentation 3D. Dans ce qui suit, nous détaillerons deux travaux récents de la tache ‘texts to 3D shapes’ :

a. ‘Generating 3D Shapes from Text using Wasserstein Generative Adversarial Networks’ : Est un travail de recherche publié en 2020 par Tiantian Wang, Xiaohu Guo et Ruigang Yang. Il propose un modèle de réseau de neurones capable de générer des formes 3D à partir de descriptions textuelles en utilisant des Réseaux Adversaires Génératifs (GAN) de Wasserstein.

Le modèle est basé sur un encodeur qui prend en entrée des descriptions textuelles et les transforme en une représentation latente. Cette représentation latente est ensuite utilisée par un décodeur pour générer des formes 3D. Le décodeur est entraîné pour générer des formes 3D qui correspond à la représentation latente, tandis que le discriminateur est entraîné pour distinguer les formes 3D générées du véritable ensemble de données de formes 3D.

Le modèle utilise également une technique appelée ‘projection sémantique’, qui permet de projeter la représentation latente d'une forme 3D existante dans l'espace de la description textuelle. Cela permet d'associer une forme 3D existante à une description textuelle correspondante, ce qui peut être utile pour la recherche d'objets en 3D ou la création de bases de données annotées.

Le modèle proposé dans ce travail a été évalué sur plusieurs ensembles de données de formes 3D et de descriptions textuelles. Les résultats montrent que le modèle est capable de générer des formes 3D qui correspondent aux descriptions textuelles avec une précision

raisonnable. De plus, la technique de projection sémantique a permis de lier des formes 3D existantes à des descriptions textuelles correspondantes avec une grande précision [32].

b. 'Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings' : Est un travail de recherche publié en 2021 par Zekun Hao, Qihang Mo, Yixin Zhuang, Yuanpei Cao et Baoquan Chen. Il propose un modèle de réseau de neurones capable de générer des formes 3D à partir de descriptions textuelles en apprenant des plongements de mots et de formes 3D conjoints.

Le modèle est basé sur une architecture en deux parties, une partie qui traite les descriptions textuelles et une partie qui traite les formes 3D. La partie qui traite les descriptions textuelles utilise un modèle de langage pré-entraîné pour apprendre des plongements de mots pour chaque mot de la description textuelle. La partie qui traite les formes 3D utilise un auto-encodeur qui apprend des plongements pour chaque forme 3D. Les plongements de mots et de formes 3D sont ensuite combinés dans un espace de plongement conjoint en utilisant une méthode de réduction de dimensionnalité.

Le modèle apprend à générer des formes 3D à partir de descriptions textuelles en utilisant une fonction de perte de reconstruction qui mesure la différence entre la forme 3D générée et la forme 3D réelle correspondante. Le modèle utilise également une fonction de perte d'adversaire pour améliorer la qualité de la génération de formes 3D.

Le modèle proposé dans ce travail a été évalué sur plusieurs ensembles de données de formes 3D et de descriptions textuelles. Les résultats montrent que le modèle est capable de générer des formes 3D qui correspondent aux descriptions textuelles avec une précision raisonnable. De plus, le modèle est capable de générer des formes 3D qui sont cohérentes avec les contraintes spatiales de la description textuelle, ce qui améliore la qualité de la génération [33].

Voici un tableau comparatif des deux travaux :

Travaux	"Generating 3D Shapes from Text using Wasserstein Generative Adversarial Networks"	"Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings"
Année de	2020	2021

publication		
Modèle utilisé	Réseaux Adversaires Génératifs de Wasserstein	Apprentissage de plongements de mots et de formes 3D conjoints
Description textuelle	Encodée en représentation latente pour la génération de formes 3D	Traitée avec un modèle de langage pré-entraîné pour apprendre des plongements de mots
Formes 3D	Générées par un décodeur à partir de la représentation latente	Générées par une fonction de perte de reconstruction et une fonction de perte d'adversaire
Techniques supplémentaires	Projection sémantique pour associer des formes 3D existantes à des descriptions textuelles correspondantes	Utilisation de contraintes spatiales pour améliorer la qualité de la génération
Evaluation	Précision raisonnable pour la génération de formes 3D correspondant aux descriptions textuelles	Précision raisonnable pour la génération de formes 3D correspondant aux descriptions textuelles et amélioration de la qualité grâce aux contraintes spatiales

Tableau 1 : Comparaison entre les travaux a et b

6.2. La génération de formes 3D à partir de texte en passant par une étape intermédiaire de transformation de texte en image:

Dans ce qui suit, nous détaillerons deux travaux récents de la tache ‘image to 3D shapes’ :

c. ‘Adversarial Synthesis of 3D Object Shape and Appearance from Sketches and Text’ par Xinyi Yang, Peng Song, Chi Zhang et Hao Zhang : Ce travail propose un modèle de génération de formes 3D à partir de croquis et de descriptions textuelles en utilisant un réseau génératif antagoniste (GAN).

Le modèle proposé comporte deux parties : le générateur et le discriminateur. Le générateur prend en entrée un croquis 2D et une description textuelle et génère une forme

3D ainsi que des textures associées. Le discriminateur, quant à lui, prend en entrée une paire constituée d'une forme 3D et de textures et détermine si cette paire a été générée à partir des données réelles ou du générateur.

Le modèle est entraîné de manière adversaire, où le générateur cherche à tromper le discriminateur en produisant des paires de forme 3D et de textures qui ressemblent à des données réelles, tandis que le discriminateur essaie de les distinguer des données réelles. L'entraînement est effectué sur un ensemble de données de croquis et de descriptions textuelles d'objets 3D provenant de la base de données ShapeNet.

Les résultats montrent que le modèle est capable de générer des formes 3D et des textures réalistes à partir de croquis et de descriptions textuelles. Le modèle peut également être utilisé pour la synthèse de formes 3D à partir d'images réelles en utilisant un réseau de transformation d'image en croquis en tant qu'étape intermédiaire.

En résumé, ce travail propose une approche novatrice pour la génération de formes 3D à partir de croquis et de descriptions textuelles en utilisant un modèle GAN entraîné de manière adversaire. Les résultats montrent que cette approche est efficace pour générer des formes 3D réalistes à partir de croquis et de descriptions textuelles [34].

d. 'Shape and Texture Generating Multi-Modal GAN for Joint Attributes and Object Stages Editing' par Xiaoyu Liu, Yiqun Lin, Yijuan Lu, Ming Tang, Jingyi Yu et Hui Huang : Ce travail propose un modèle de génération de formes 3D et de textures à partir d'attributs textuels et de stades d'objet spécifiques.

Le modèle proposé est un GAN multi-modal qui génère simultanément des formes 3D et des textures à partir d'une représentation latente. Cette représentation latente est construite en combinant un vecteur de bruit aléatoire et un vecteur d'attributs textuels qui décrivent les caractéristiques de l'objet à générer.

En outre, le modèle est également capable de générer des formes 3D et des textures à différents stades de l'objet. Cela signifie que le modèle peut générer des objets qui passent par différents stades de développement, tels que la croissance d'une plante ou l'évolution d'un personnage 3D.

Le modèle est entraîné sur un ensemble de données de formes 3D et de textures associées à partir de la base de données ShapeNet. Les résultats montrent que le modèle est capable de générer des objets réalistes avec différentes formes 3D et textures en utilisant des attributs textuels spécifiques et des stades d'objet.

En résumé, ce travail propose un modèle de génération de formes 3D et de textures multi-modal qui peut générer des objets à partir d'attributs textuels spécifiques et de stades d'objet. Les résultats montrent que ce modèle est efficace pour générer des objets réalistes avec des formes 3D et des textures différentes en utilisant des attributs textuels spécifiques et des stades d'objet [35].

Voici un tableau comparatif des deux travaux :

Travaux	"Adversarial Synthesis of 3D Object Shape and Appearance from Sketches and Text"	"Shape and Texture Generating Multi-Modal GAN for Joint Attributes and Object Stages Editing"
Année de publication	2020	2021
Approche	GAN	GAN
Données d'entrée	Textes et croquis	Textes, attributs et stades d'objet
Étape intermédiaire	Image 2D	Non
Données de sortie	Formes 3D et textures	Formes 3D et textures
Type de génération	Synthèse adversariale	Synthèse adversariale multi-modale
Résultats	Les résultats montrent que le modèle peut générer des objets réalistes à partir de textes et de croquis	Les résultats montrent que le modèle peut générer des objets réalistes avec des formes 3D et des textures différentes en utilisant des attributs textuels spécifiques et des stades d'objet
Avantages	Étape intermédiaire pour aider à la génération de formes 3D	Génération multi-modale pour une génération plus précise

Inconvénients	Limitation à la génération de formes simples	Requiert des données d'attributs et de stades spécifiques

Tableau 2 : Comparaison entre les travaux c et d

7. Conclusion :

Dans ce chapitre nous avons présenté quelques notions de bases de l'apprentissage automatique et de l'apprentissage profond, ainsi, on a cité quelques exemples d'architectures de réseaux de neurones et on a conclu ce chapitre par des travaux réalisé dans le domaine de 'text to shape'.

Les modèles de génération de formes 3D à partir de descriptions textuelles présentés sont fortement dépendants des données d'entraînement utilisées ou bien ne sont capables que de générer de formes simples. Si les données d'entraînement ne couvrent pas une grande variété de formes et de descriptions, le modèle peut avoir du mal à générer des formes 3D précises et diversifiées à partir de nouvelles descriptions textuelles. De plus, aucun d'eux ne fait appel aux LLM (Large Language Models) pour capturer le sens de la description textuelle en entrée. Ils se limitent tous à du word2vect traditionnel.

Dans notre travail, on essaiera de mettre en œuvre une architecture générative utilisant comme entrée une vectorisation basée sur les LLM. Le prochain chapitre sera consacré à la partie conception ou nous allons exposer l'architecture globale de notre approche 'text to shape'.

Chapitre 2 : Construction d'un modèle d'apprentissage pour la génération des formes 3D à partir de texte à base de CGAN et Auto-Encoder

1. Introduction :

Ce chapitre présente la section conceptuelle où nous expliquerons l'approche adoptée dans notre projet pour générer des formes 3D à partir de texte. Dans un premier temps, nous décrirons l'architecture globale de notre travail, puis nous aborderons la première partie de notre tâche, qui consiste à convertir un texte en un vecteur numérique. Nous discuterons du modèle **'all-MiniLM-L6-v2'** utilisé pour l'encodage du texte. Enfin, nous traiterons de la deuxième partie, qui concerne la génération de formes 3D, en décrivant les composants de nos modèles et les fonctions de perte utilisées.

2. Architecture globale :

Notre projet se compose de deux principales parties :

- La conversion des descriptions textuelles en vecteurs numériques :

  Pour cette étape, nous avons utilisé le modèle pré-entraîné "all-MiniLM-L6-v2" [36], qui est une combinaison de BERT et SBERT dans son approche. Ce modèle prend une description textuelle en entrée et la convertit en un vecteur numérique dense sous forme de tableau NumPy.

- La génération de formes 3D à partir de descriptions textuelles :

  Dans cette partie, on a exploité deux modèles : CGAN et Auto-encoder. Pour le CGAN, on a l'utilisé avec le vecteur de bruit et sans vecteur de bruit.

L'architecture du CGAN comprend deux composants principaux : le générateur et le discriminateur.

Le générateur utilise un vecteur qui résulte de la fusion du vecteur de bruit aléatoire (noise) avec l'embedding sémantique (la condition) pour le modèle CGAN avec noise, et seulement l'embedding sémantique pour le modèle CGAN sans noise. Dans les deux approches, l'embedding sémantique est converti préalablement en un tenseur PyTorch de type float. Ensuite, le générateur produit un tenseur PyTorch représentant la forme 3D correspondante à la description textuelle donnée. Le discriminateur analyse ensuite si cette forme générée est réelle ou fausse (real or fake). Pour cela, le discriminateur prend en entrée la concaténation de l'embedding sémantique avec la forme générée par le générateur, puis prédit la validité de l'image générée (réelle ou générée). Ce processus ce répète jusqu'a la fin de l'entrainement. C'est ainsi que le générateur et le discriminateur s'entraînent de manière adversariale à l'aide de la fonction de perte 'MSELoss', qui permet de mettre à jour les poids respectifs de générateur et de discriminateur.

Notre modèle Auto-Encoder suit une structure d'encodeur-décodeur pour la génération de formes 3D.

L'encodeur comprend des couches de convolution 3D qui réduisent progressivement la dimensionnalité de l'image d'entrée, suivies d'une couche linéaire qui produit un vecteur latent de plus faible dimension. Le décodeur, quant à lui, utilise des couches linéaires pour augmenter la dimension du vecteur latent, suivi de couches de convolution transposées 3D qui reconstruisent l'image en utilisant les informations du vecteur latent. La dernière couche de convolution transposée utilise une fonction d'activation sigmoïde pour générer les valeurs d'image dans la plage (0, 1). En utilisant cette structure, le modèle est capable de prendre une image en entrée, extraire ses caractéristiques clés via l'encodeur, et générer une image reconstruite à partir du vecteur latent via le décodeur.

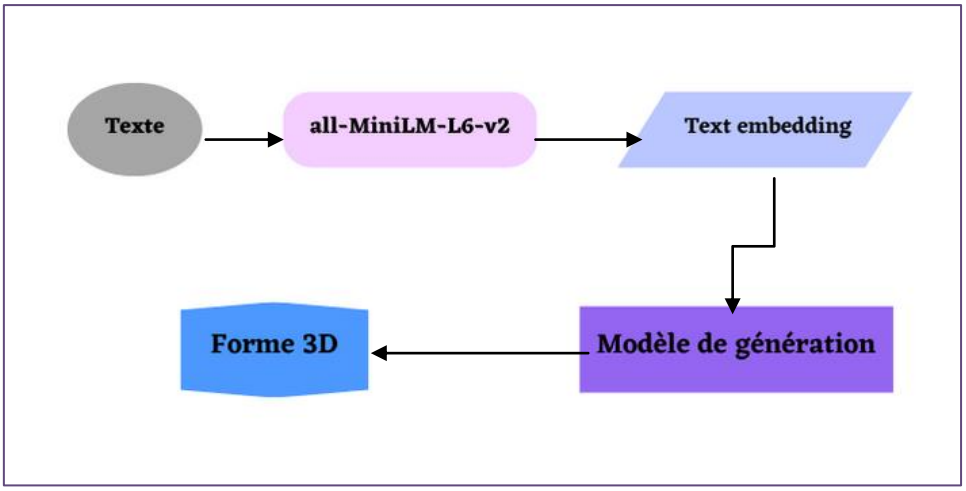

Figure 13 : Architecture globale du travail

2.1. Conversion de nos descriptions textuelles en vecteurs numériques :

Le modèle 'all-MiniLM-L6-v2' a été utilisé pour les phrases (descriptions textuelles) pour obtenir des embeddings de haute qualité qui capturent les informations sémantiques et contextuelles du texte.

L'avantage avec un tel modèle est qu'il intègre une couche de prétraitement incorporée, ce qui réduit la complexité de notre tâche.

Cette méthode prend en entrée une liste de phrase(s) et renvoie une matrice de vecteur(s) numérique(s) dense(s) correspondant(s). La taille des vecteurs générés est 384.

2.1.1. Le modèle de plongement de texte 'all-MiniLM-L6-v2' : Est un modèle de plongement de phrases pré-entraîné par l'équipe de recherche SentenceTransformers. C'est une variante du modèle SentenceTransformer qui utilise l'architecture MiniLM avec 6 couches pour générer des embeddings de phrases. Cette architecture est une version réduite du modèle de langage naturel BERT, qui a été spécialement optimisé pour une utilisation dans les tâches de compréhension de phrases et de classification de texte.

'all-MiniLM-L6-v2' utilise une technique appelée "Transformer-based architecture" pour convertir les phrases en vecteurs numériques. Cette architecture est basée sur des réseaux de neurones profonds qui sont capables de capturer les relations sémantiques entre les mots et les phrases. Lorsque le modèle encode les phrases, il utilise des techniques d'attention pour donner plus d'importance aux parties de la phrase qui sont les plus pertinentes pour la signification globale de la phrase [36].

2.1.2. Le choix du modèle "all-MiniLM-L6-v2" :

Le modèle "all-MiniLM-L6-v2" présente des caractéristiques et avantages qui le rendent un choix approprié :

1. Capacité d'encodage : Le modèle est spécifiquement conçu pour encoder des séquences de texte en vecteurs numériques. Il peut capturer les informations sémantiques et contextuelles dans les descriptions textuelles et les représenter de manière compréhensible pour nos modèles de génération de formes 3D.
2. Capacité et Performances élevées de traitement du langage naturel : Le modèle "all-MiniLM-L6-v2"est basé sur l'architecture MiniLM, qui est spécifiquement conçue pour une meilleure performance de langage. Il est pré-entraîné sur une vaste quantité de données textuelles provenant de diverses sources, ce qui lui confère de solides compétences en compréhension de texte. Il peut capturer les relations sémantiques entre les mots et les concepts, ce qui est crucial pour comprendre et représenter les descriptions textuelles de formes 3D de manière précise.
3. Grande disponibilité et facilité d'utilisation : Le modèle "all-MiniLM-L6-v2" est largement accessible et peut être utilisé via des bibliothèques de traitement du langage naturel. Il est donc facile à intégrer dans notre projet.
4. Taille du modèle : "all-MiniLM-L6-v2" est un modèle relativement compact par rapport à d'autres modèles de langage plus volumineux. Cela signifie qu'il nécessite moins de ressources informatiques et de mémoire pour fonctionner.

Les figures 15 et 16 sont des captures du site officiel ‘sbert.net’ qui représentes les spécifications principales du modèle ‘all-MiniLM-L6-V2’ [36].

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 14 : Caractéristiques du modèle all-MiniLM-L6-V2 [36]

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 15 : Caractéristiques du modèle all-MiniLM-L6-V2 [36]

2.2. Conversion des vecteurs en formes 3D :

Dans notre travail nous avons opté pour deux modèles distincts : l'un se basant sur les CGAN et le second sur les auto-encoder.

2.2.1. Le choix d'une architecture basée sur CGAN :

Voici certaines raisons principales qui nous ont poussés à choisir le CGAN:

- Modélisation conditionnelle : Le modèle CGAN est capable de générer des échantillons conditionnés sur une entrée spécifique, dans notre cas, les descriptions textuelles. Cela signifie que nous pouvons fournir une description textuelle en tant que condition pour

guider la génération du modèle 3D. Cette approche permet de créer des formes 3D qui sont cohérentes avec les informations sémantiques présentes dans les descriptions.

- Capture de la variabilité : Les formes 3D peuvent avoir différentes variations et styles en fonction des descriptions textuelles fournies. Le modèle CGAN peut capturer cette variabilité en générant des formes 3D diverses et uniques correspondant aux différentes descriptions textuelles.
- Génération créative : Le modèle CGAN peut générer des formes 3D de manière créative et exploratoire en utilisant les descriptions textuelles comme guide. Il peut produire des résultats inattendus et originaux, ce qui peut être bénéfique pour générer des formes 3D intéressantes et novatrices.

Possibilité d'amélioration : Les modèles CGAN sont continuellement améliorés et affinés par la communauté de recherche. Il existe de nombreuses variantes et extensions du modèle CGAN qui intègrent de nouvelles techniques et architectures pour améliorer les performances de génération. Nous pouvons donc bénéficier de ces développements pour obtenir des résultats de génération de formes 3D plus précis et de meilleure qualité au fil du temps.

2.2.2. Le choix d'une architecture basée sur Auto-Encoder :

Voici quelques raisons principales qui nous ont poussés à choisir l'Auto-Encoder :

- Apprentissage non supervisé : Le modèle auto-encodeur est capable d'apprendre des représentations latentes utiles à partir de données non étiquetées. Cela permet d'effectuer une compression ou une réduction de dimension des données sans nécessiter des étiquettes de sortie.
- Extraction automatique des caractéristiques : L'auto-encodeur est capable d'extraire automatiquement les caractéristiques les plus discriminantes des données en comprimant l'entrée dans une représentation latente de dimension réduite. Ces caractéristiques apprises peuvent être utilisées pour la classification, la génération ou d'autres tâches.
- Capacité de reconstruction : Le modèle auto-encodeur est conçu pour reconstruire les données d'entrée à partir de la représentation latente. Cela signifie qu'il est capable de capturer les principales caractéristiques des données et de les restituer avec une certaine fidélité. Cette capacité de reconstruction peut être utile pour la détection d'anomalies ou la récupération des données corrompues.

- Apprentissage de représentations compressées : L'auto-encodeur peut être utilisé pour apprendre des représentations compressées des données. En réduisant la dimension de l'entrée, il permet de stocker et de traiter les données de manière plus efficace, tout en préservant les informations importantes.
- Génération de données : En utilisant le décodeur du modèle auto-encodeur, il est possible de générer de nouvelles données similaires à celles de l'ensemble d'entraînement. Cela peut être utilisé pour la génération de contenu créatif, la complétion de données manquantes ou la simulation de scénarios divers.
- Robustesse aux variations mineures : L'auto-encodeur est capable de capturer les variations mineures dans les données et de les représenter dans la représentation latente. Cela rend le modèle robuste aux petites variations et bruits présents dans les données.
- Architecture flexible : Le modèle auto-encodeur peut être adapté à différentes tâches en ajustant l'architecture du réseau, comme le nombre de couches, les types de couches ou les dimensions des représentations latentes. Il offre donc une certaine flexibilité pour s'adapter à différents problèmes.

2.2.3. Les composants de nos modèles CGAN :

Un modèle CGAN est composé de deux parties : Un générateur et un critique (discriminateur) :

2.2.3.1. Générateur :

Dans un cas général, la partie génératrice du GAN utilisé un vecteur de bruit (noise) pour générer de l'information, un CGAN y rajoute une condition en plus à ce vecteur. Dans notre cas, la condition en entrée va être la description textuelle représentée sous sa forme vectorielle obtenue précédemment par 'all-MiniLM-L6-V2'. Nous allons proposer deux variantes, un modèle exploitant le vecteur de bruit et un sans vecteur de bruit.

- **Avec vecteur de bruit (Noisy CGAN) :** Ce générateur est un réseau de neurones artificiels conçu pour générer des images de résolution 32, de la forme (4 canaux, 32, 32,32), de taille (4*32*32*32), à partir d'un vecteur de bruit aléatoire de taille 384 et d'un vecteur d'embedding de texte de taille 384 en entrée, en suivant une séquence de couches linéaires, avec de normalisation de batch et de fonctions d'activation 'LeakyReLU'. Il est une classe qui hérite de la classe 'nn.Module' de la bibliothèque 'PyTorch' de Python.

    - **Structure :**

La structure de ce dernier est définie par une séquence de blocs linéaires suivis d'une normalisation de batch et d'une fonction d'activation ‘LeakyReLU’. Chaque bloc est conçu pour augmenter progressivement la complexité de la sortie en augmentant le nombre de neurones et en appliquant une normalisation de batch.

Couche d'entrée : Le générateur commence par une couche linéaire de taille 384+384 qui prend en entrée la concaténation du vecteur de bruit aléatoire (noise) et du vecteur d'embedding de texte et donne une sortie de taille 128. Elle est suivie d’une fonction d'activation ‘LeakyReLU’ (slope de 0.2).

Bloc 1 : Ce bloc est constitué d'une couche linéaire de taille 128 qui génère en sortie un tenseur de taille 256. Il est suivi d'une couche de normalisation de batch avec un facteur de normalisation de 0.8 et d'une fonction d'activation ‘LeakyReLU’ (slope de 0.2).

Bloc 2 : Ce bloc est constitué d'une couche linéaire de taille 256 qui donne en sortie un tenseur de taille 512. Il est également suivi d'une couche de normalisation de batch et d'une fonction d'activation ‘LeakyReLU’.

Bloc 3 : Ce bloc est constitué d'une couche linéaire de taille 512 qui génère un tenseur de taille 1024 en sortie. Il comprend également une couche de normalisation de batch et une fonction d'activation ‘LeakyReLU’.

Couche de sortie : La dernière couche du générateur est une couche linéaire de taille 1024 qui produit une sortie de taille (4*32*32*32) en sortie. Cette couche n'est pas suivie d'une normalisation de batch. La sortie de cette couche est ensuite passée à travers une fonction d'activation ‘Tanh’.

Cette structure du générateur est résumée dans la figure 17 :

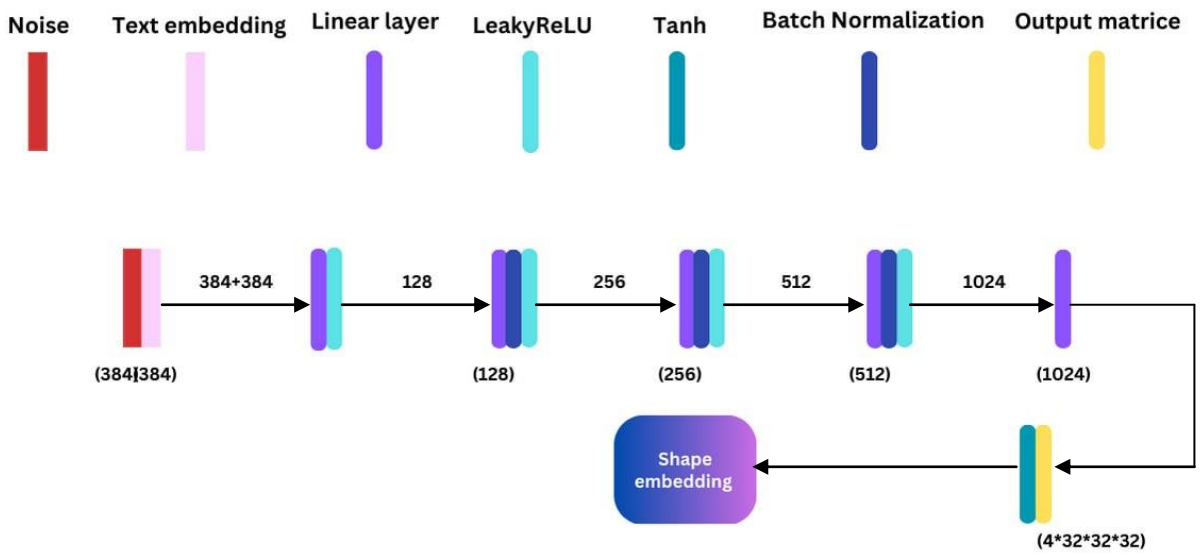

Figure 16 : Architecteur globale du générateur, CGAN avec noise

- **Fonctionnement :**

Tout d'abord, le générateur concatène le noise avec l'embedding de texte en utilisant la fonction 'torch.cat()'. Cette opération combine les informations des deux vecteurs, permettant ainsi d'introduire à la fois de la variabilité et des caractéristiques spécifiques à l'étiquette de classe dans le processus de génération.

Ensuite, le vecteur résultant est transmis à travers les différentes couches du générateur. Chaque couche est responsable de transformer les caractéristiques de l'entrée et d'ajouter de la complexité à la sortie.

Le générateur utilise des couches linéaires (Fully Connected) qui effectuent une transformation linéaire du vecteur d'entrée en multipliant les poids appropriés. Les poids sont appris pendant l'entraînement du générateur afin d'optimiser la génération des images souhaitées.

Les fonctions d'activation 'LeakyReLU' introduisent de la non-linéarité dans le processus. Elles sont utilisées pour permettre au générateur d'apprendre des relations complexes entre les caractéristiques d'entrée et de sortie.

La normalisation de batch (BatchNorm1d) est appliquée après certaines couches pour normaliser les activations et accélérer la convergence de l'apprentissage. Elle contribue également à stabiliser l'entraînement du générateur.

Finalement, la dernière couche linéaire est suivie d'une fonction d'activation ‘Tanh’, qui restreint la plage des valeurs de la sortie finale entre -1 et 1. Cela permet d'obtenir une image générée avec des valeurs de voxel dans une plage utilisable.

La sortie finale est ensuite remodelée en une image en utilisant la fonction ‘view()’ de PyTorch, en spécifiant les dimensions souhaitées de l'image. Cette opération permet de donner à la sortie la forme souhaitée pour représenter une image.

- **Sans vecteur de bruit (Noiseless CGAN) :**

Ce générateur a exactement la même structure et le même fonctionnement que le générateur du CGAN avec noise, la seule différence est l’entrée de ce générateur. Ce dernier prend en entrée le vecteur d’embedding seulement.

2.2.3.2. Discriminateur : Il a la même structure et fonctionnement pour les deux architectures CGAN.

Le discriminant est un réseau de neurones qui prend en entrée à la fois l'image générée par le générateur de taille 4*32*32*32 et l'embedding de texte correspondant de taille 384 pour prédire si l'image générée est réelle ou fausse. Il est de même une classe héritant de la classe ‘nn.Module’ de la bibliothèque ‘PyTorch’.

- **Structure :**

La structure du discriminateur est définie par une séquence de couches linéaires avec des couches de dropout et des fonctions d'activation ‘LeakyReLU’.

Couche d'entrée : La couche d'entrée est une couche linéaire qui prend deux entrées concaténées, la première étant la forme générée de taille 4*32*32*32 et la seconde l'embedding de texte de taille 384, et produit un tenseur de sortie de taille 512. Cette couche est suivie d'une fonction d'activation ‘LeakyReLU’ (slope de 0.2).

Couche cachée 1 : C'est une couche linéaire de taille 512 qui prend en entrée le tenseur de sortie de la couche d'entrée et produit un tenseur de sortie de taille 512. Cette couche est également suivie d'une fonction d'activation 'LeakyReLU' et d'une couche de dropout avec une probabilité de 0,4 qui désactive aléatoirement certains neurones afin de réduire le sur-apprentissage.

Couche cachée 2 : C'est une couche linéaire de taille 512 qui prend en entrée le tenseur de sortie de la première couche et produit un tenseur de sortie de taille 512. Cette couche est également suivie d'une fonction d'activation 'LeakyReLU' et d'une couche de dropout avec une probabilité de 0,4.

Couche de sortie : La couche de sortie est une couche linéaire de taille 512 qui prend en entrée le tenseur de sortie de la deuxième couche et produit un tenseur de sortie de taille 1, qui représente la validité de l'image générée par le générateur.

En résumé, la structure du discriminateur est présentée dans la figure 18 :

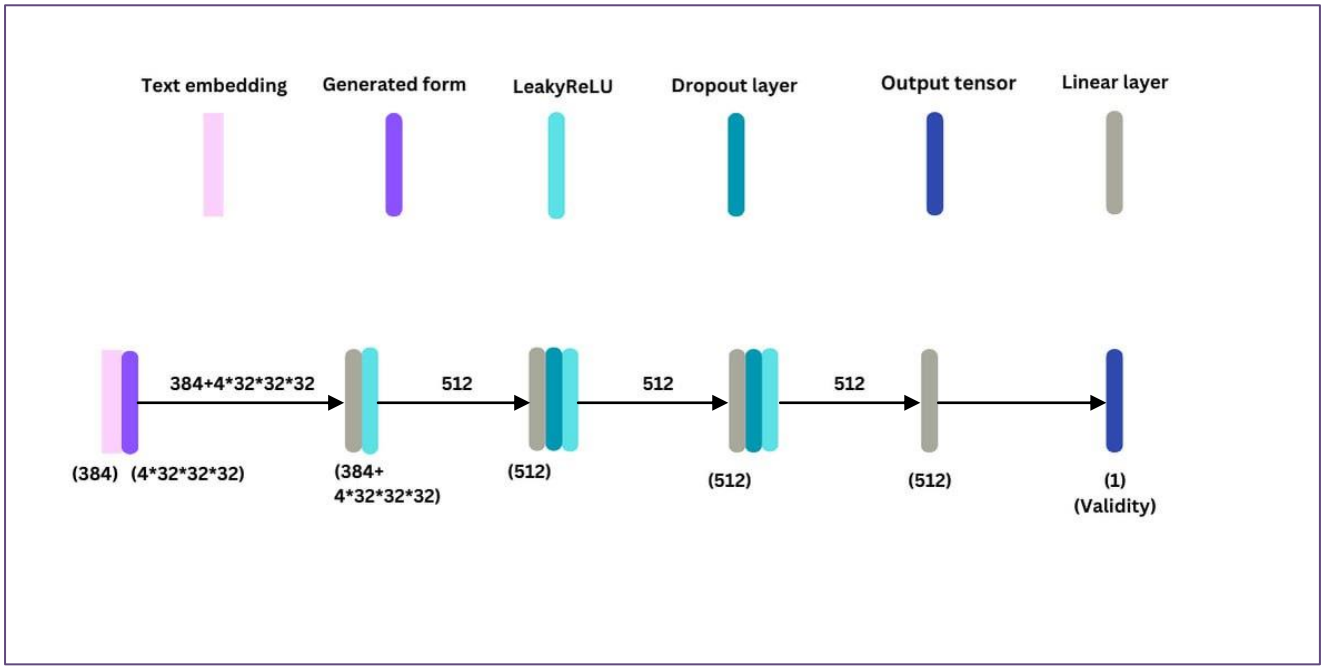

Figure 17 : Architecture globale du discriminateur

- **Fonctionnement :**

Le discriminant prend en entrée un tenseur de taille 384+4*32*32*32. Ce tenseur est ensuite propagé à travers les différentes couches du discriminant pour produire une sortie de taille 1 qui représente si l'image générée est réelle ou fausse.

2.2.3.3. Fonctions de perte :

Voici les fonctions de perte utilisées dans nos modèles CGAN, lors de l'entrainement :

- Fonction de perte du générateur (Generator Loss) : La fonction de perte utilisée par le générateur est la 'Mean Squared Error Loss (MSELoss)', qui mesure la similarité entre les images générées et les images réelles en calculant la moyenne des erreurs au carré entre les prédictions et les cibles. Les prédictions correspondent à la sortie du discriminateur pour les images générées, tandis que les cibles sont toutes définies à 1 (pour indiquer que ce sont de vraies images). L'objectif du générateur est de minimiser cette perte pour tromper le discriminateur.
- Fonction de perte du discriminateur (Discriminator Loss) : La fonction de perte utilisée par le discriminateur est également la 'Mean Squared Error Loss'. Le discriminateur vise à distinguer les images réelles des images générées par le générateur. Ainsi, la perte du discriminateur mesure à la fois la capacité du discriminateur à classifier correctement les images réelles et sa capacité à classifier les images générées comme fausses. Elle est calculée par la moyenne des erreurs au carré entre les prédictions et les cibles. Les prédictions pour les images réelles sont comparées aux cibles de 1, et les prédictions pour les images générées sont comparées aux cibles de 0.

- **La fonction de perte Mean Squared Error Loss (MSELoss) :** Est une mesure couramment utilisée pour évaluer la différence entre les valeurs prédites et les valeurs réelles dans les tâches de régression. Elle calcule la moyenne des erreurs au carré entre les prédictions et les cibles.

Dans le contexte des réseaux de neurones, la MSELoss est définie comme suit : Soit 'y_pred' la valeur prédite par le modèle et 'y_true' la valeur réelle (cible). La MSELoss est calculée comme suit : **MSELoss(y_pred, y_true) = (1/n) * ∑(y_pred[i] - y_true[i])^2** , où n est le nombre total d'échantillons dans le batch (batch size).

Plus la valeur de MSELoss est faible, plus les prédictions du modèle sont similaires aux valeurs réelles [40].

2.2.4. Les composants de notre modèle Auto-Encoder :

Notre modèle auto-encodeur est un ‘diffusion model’. Il prend en entrée une image 3D et tente de reconstruire une version similaire de l'image en sortie.

Son architecture est composée de deux parties principales : l'encodeur et le décodeur :

2.2.4.1. L’encodeur :

L'encodeur est responsable de la compression de l'image d'entrée en une représentation latente de dimension réduite.

L'encodeur est composé de couches convolutionnelles 3D suivies de fonctions d'activation ReLU. Ces couches sont conçues pour extraire les caractéristiques importantes des images en réduisant progressivement leur taille. Les couches de convolution sont utilisées pour capturer les motifs spatiaux dans les images.

- **Structure :**

1. Une couche de convolution 3D avec un noyau de taille 3x3x3, qui transforme l'image d'entrée avec "4" canaux en une représentation avec "25" canaux.
2. Une fonction d'activation ReLU, qui introduit une non-linéarité dans la sortie de la première couche de convolution.
3. Une autre couche de convolution 3D qui transforme la sortie précédente avec 25 canaux en une représentation avec "25 * 2" canaux. Cette couche réduit la taille spatiale de la représentation en la divisant par 2 grâce à un stride de 2.
4. Une autre couche de convolution 3D qui transforme la sortie précédente avec "25 * 2" canaux en une représentation avec "25 * 4" canaux. Cette couche réduit à nouveau la taille spatiale de la représentation par un facteur de 2.
5. Une dernière couche de convolution 3D qui transforme la sortie précédente avec "25 * 4" canaux en une représentation avec "25 * 8" canaux. Cette couche réduit la taille spatiale de la représentation d'un facteur de 2.

6. La sortie des couches de convolution est aplatie en un vecteur unidimensionnel à l'aide de la fonction Flatten.

7. Un réseau de neurones fully connected (Linear) qui prend en entrée le vecteur aplati et le transforme en une représentation de taille "25".

8. Une dernière fonction d'activation ReLU qui introduit à nouveau de la non-linéarité dans la sortie du réseau fully connected.

- **Fonctionnement :**

L'encodeur est responsable de la compression de l'entrée, qui est une forme 3D, en une représentation latente de dimension réduite. Voici comment il fonctionne :

- L'entrée, représentée par une forme 3D avec un certain nombre de canaux (4), est passée à travers des couches de convolutions 3D. Chaque couche utilise un noyau de taille 3x3x3 et une fonction de padding pour maintenir les dimensions.
- Les couches de convolutions captent les caractéristiques de l'entrée à différentes échelles et les transforment en un ensemble de cartes de caractéristiques avec un nombre croissant de canaux (25 * 2, 25 * 4, 25* 8).
- La dernière couche de convolutions réduit la taille spatiale de l'image tout en maintenant le nombre de canaux élevé, afin d'obtenir une représentation latente dense.
- Ensuite, l'image est aplatie pour obtenir une représentation linéaire, puis passe à travers une couche linéaire qui réduit encore davantage la dimension vers 25, la dimension de la représentation latente.
- Une fonction d'activation ReLU est appliquée à la couche linéaire pour introduire de la non-linéarité et obtenir la représentation latente finale.

2.2.4.2. Le décodeur :

Le décodeur effectue l'opération inverse de l'encodeur pour reconstruire l'image à partir de la représentation latente.

- **Structure :**

1. Un réseau de neurones fully connected (Linear) qui prend en entrée la représentation latente de taille "25" et la transforme en un vecteur de taille "25 * 8 * 32 ** 3".

2. Une fonction d'activation ReLU qui introduit de la non-linéarité dans la sortie du réseau fully connected.

3. Une opération Unflatten qui transforme le vecteur en un tenseur de taille (25 * 8, 32, 32, 32).

4. Une série de couches de transposition de convolution (ConvTranspose3d) qui effectuent des opérations inverses des couches de convolution de l'encodeur. Ces couches augmentent progressivement la taille spatiale de la représentation.

5. Chaque couche de transposition de convolution est suivie d'une fonction d'activation ReLU, à l'exception de la dernière couche.

6. La dernière couche de transposition de convolution transforme la représentation avec "25" canaux en une représentation avec "4" canaux.

7. La dernière couche utilise la fonction d'activation 'sigmoid' pour produire une sortie dans la plage [0, 1], représentant une image reconstruite.

- **Fonctionnement :**

Le décodeur prend la représentation latente en entrée et tente de reconstruire la forme 3D d'origine. Voici comment il fonctionne :

- La représentation latente est d'abord passée à travers une couche linéaire qui projette la dimension de 25 vers 25 * 8 *32^3 , correspondant à la taille spatiale de l'image 3D.
- Une fonction d'activation ReLU est appliquée pour introduire de la non-linéarité.
- Ensuite, une opération de remodelage est utilisée pour transformer la sortie linéaire en une forme 3D avec 25* 8 canaux et une taille spatiale de 32.
- Les couches de convolutions transposées sont utilisées pour inverser les opérations de convolution effectuées par l'encodeur. Ces couches augmentent la taille spatiale de l'image tout en réduisant le nombre de canaux.
- Chaque couche de convolution transposée est suivie d'une fonction d'activation ReLU pour introduire de la non-linéarité dans la génération de l'image.

- Enfin, la dernière couche de convolution transposée a un nombre de canaux égal à 4, correspondant au nombre de canaux de la forme 3D d'origine. Une fonction d'activation ‘sigmoïde’ est appliquée pour normaliser les valeurs des pixels de sortie entre 0 et 1.

La structure générale de notre Auto-Encoder est schématisée dans la figure 19 :

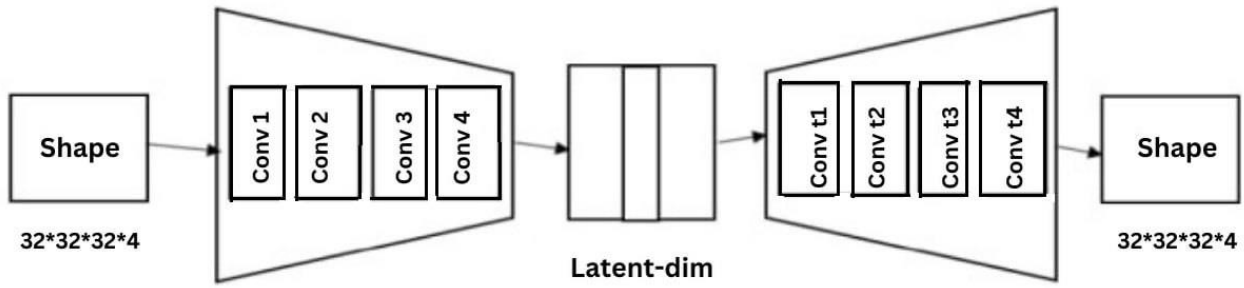

Figure 18 : Architecture générale de notre Auto-Encoder

3. Conclusion :

Ce chapitre pose les bases conceptuelles nécessaires de notre projet, en fournissant une vision claire de la méthodologie adoptée pour convertir les descriptions textuelles en formes 3D.

Dans le chapitre suivant, nous parlerons des outils utilisés dans le long de notre processus, avec les résultats obtenues et l’évaluation de notre travail.

Chapitre 3 : Réalisation

1. Introduction :

Tout au long de notre processus, nous avons utilisé différentes ressources et obtenu plusieurs résultats. Dans ce chapitre, nous présenterons une brève définition de ces outils qui nous ont aidés dans la réalisation de notre travail, ainsi que les résultats que nous avons obtenus. Nous procéderons ensuite à une évaluation et une discussion sur ces résultats. En conclusion, nous exposerons notre application desktop qui illustre notre projet.

2. Outils utilisés :

- Google Colab : Est un service de notebook collaboratif basé sur le cloud, qui permet aux utilisateurs d'écrire, d'exécuter et de partager du code Python. Il fournit une plate-forme de traitement parallèle gratuite avec des GPU Tesla pour des calculs accélérés et des bibliothèques populaires préinstallées. Les notebooks peuvent être partagés et collaborer en temps réel avec d'autres utilisateurs via Google Drive.

- Python : Est un langage de programmation polyvalent, utilisé dans de nombreuses applications y compris l'intelligence artificiel. Il est particulièrement utile pour effectuer des tâches complexes telles que l'apprentissage automatique et la visualisation de données allant des graphiques de base aux tracés 3D plus complexes. Il dispose également d'une grande variété de bibliothèques qui permettent aux développeurs de travailler plus efficacement.

-PyTorch : Est une bibliothèque open source de calculs tensoriels largement utilisée dans le domaine de l'intelligence artificielle et de l'apprentissage automatique. Elle a été développée par Facebook AI Research. Elle facilite la création et l'entraînement de réseaux de neurones.

- Pickle : Est une bibliothèque Python utilisée pour la sérialisation et la désérialisation d'objets Python. Elle permet de convertir des objets en une représentation binaire, facilitant ainsi leur stockage, leur partage et leur chargement ultérieur.

- NumPy : Abréviation de "Numerical Python", est une bibliothèque Python open source qui fournit des structures de données performantes pour la manipulation de tableaux multidimensionnels appelés ndarrays (n-dimensional arrays), ainsi que des fonctions mathématiques avancées pour effectuer des opérations numériques efficaces.

- SentenceTransformers : Est une bibliothèque puissante pour l'apprentissage de représentations de phrases dans le domaine du traitement du langage naturel. Elle se base sur les modèles de Transformer. L'objectif principal de "sentence_transformers" est de convertir des phrases en vecteurs denses, également appelés embeddings, qui capturent les informations sémantiques et contextuelles des phrases.

- NRRD : Est un format de fichier conçu pour stocker des données volumétriques en 3D ou en dimensions supérieures, accompagné d'informations sur l'espace de données et les voxels. Ce format est utilisé pour stocker différents types de données volumétriques, tels que des images médicales ou des simulations numériques. Il peut être manipulé en utilisant la bibliothèque Python 'Pynrrd'.

- Visual Studio Code : Est un éditeur de code multiplateforme qui prend en charge une large gamme de langages de programmation et de frameworks. Il offre une interface utilisateur intuitive, une personnalisation flexible et des outils de débogage intégrés pour faciliter le développement de logiciels de qualité.

- Tkinter : Est une bibliothèque graphique en Python utilisée pour créer des applications de bureau. Elle offre des widgets et des outils pour construire des interfaces utilisateur interactives. Tkinter est intégrée à la bibliothèque standard de Python, ce qui facilite son utilisation.

- Matplotlib : Est une bibliothèque de visualisation de données en Python qui permet de créer différents types de graphiques tels que des lignes, des barres, des histogrammes, des nuages de points, etc.

- mpl_toolkits.mplot3d : Est un module de la bibliothèque Matplotlib qui permet de créer des graphiques en 3D. Ce module est souvent utilisé pour représenter des données volumineuses ou des ensembles de données complexes nécessitant une visualisation en trois dimensions.

3. DataSet :

Dans notre étude, nous avons fait usage du jeu de données "ShapeNet" : C'est une collection de données volumineuse utilisée dans le domaine de la vision par ordinateur pour la reconnaissance et la génération de formes 3D. Il est largement utilisé pour l'apprentissage automatique et l'intelligence artificielle. ShapeNet comprend une vaste bibliothèque de modèles 3D réalistes, représentant différentes catégories d'objets, tels que des meubles, des véhicules, des animaux, des objets ménagers, etc. Ces modèles 3D sont annotés et organisés de manière hiérarchique selon les catégories et les sous-catégories d'objets.

Dans notre cas, nous avons utilisé deux sous-ensembles spécifiques du dataset ShapeNet :

- Le premier est un dataset contenant 15038 fichiers NRRD de formes 3D de tables et de chaises, avec une résolution de 32.

Initialement, nous avions envisagé une résolution de 128. Cependant, en raison de limitations de mémoire RAM, nous avons utilisé la résolution 32 des formes 3D. Cette résolution plus basse offre une meilleure gestion de la mémoire et facilite le traitement des données 3D dans notre environnement de travail.

Dans la figure 19, un exemple qui illustre les différentes résolutions des images de formes 3D disponibles sur le dataset ShapeNet.

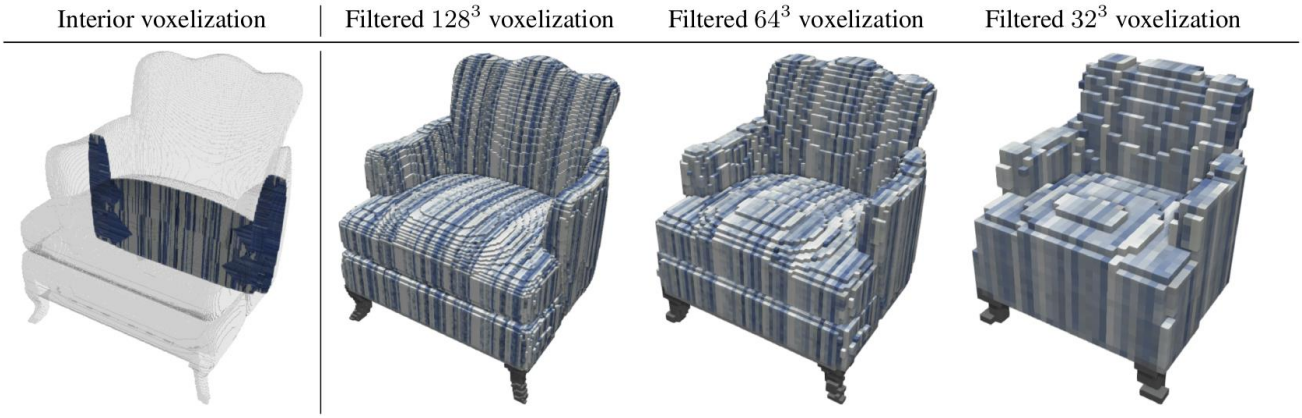

Figure 19 : Les différentes résolutions des images de Formes 3D sur le dataset ShapeNet [41]

- Le deuxième dataset que nous avons utilisé est un dataset correspondant contenant des descriptions textuelles de ces formes 3D. Ces descriptions textuelles fournissent des informations sur les caractéristiques, les dimensions, les matériaux, les styles ou toute autre propriété pertinente des formes 3D de tables et de chaises présentes dans le premier dataset. Ce dataset contient plus de 76000 descriptions en langage naturel, avec en moyenne 5 descriptions par forme.

  La figure 20 présente un exemple de descriptions textuelles avec leurs formes 3D correspondantes.

Figure 20 : Exemples de formes 3D avec leurs descriptions textuelles [41]

4. Métriques d'évaluation : On a choisie deux métriques pour l'évaluation de nos modèles : 'occupancy' et 'color/occupancy'.

4.1. La matrice de confusion : Est une représentation tabulaire qui permet de récapituler les performances d'un modèle. Elle présente les prédictions du modèle par rapport aux valeurs réelles des données.

- Prédiction du modèle :
  - Positive -> True Positive (TP), False Positive (FP),
  - Negative -> False Negative (FN), True Negative (TN).
- Les termes dans la matrice de confusion ont les significations suivantes :
  - True Positive (TP) : Le modèle a correctement prédit une instance positive.
  - False Positive (FP) : Le modèle a incorrectement prédit une instance positive.
  - False Negative (FN) : Le modèle a incorrectement prédit une instance négative.
  - True Negative (TN) : Le modèle a correctement prédit une instance négative.

4.2. Occupancy : Est une métrique utilisée pour évaluer la précision avec laquelle les voxels générés par un modèle de deep learning correspondent aux voxels réels d'une forme cible dans un environnement en 3D. L'occupation fait référence à la présence ou à l'absence d'un objet dans un voxel spécifique. Cette métrique permet de mesurer à quel point les voxels générés reproduisent fidèlement l'occupation des voxels réels, indépendamment de la couleur ou d'autres attributs.

On a calculé l'occupancy en utilisant les trois métriques de performance suivantes :

- **Recall (rappel) :** Elle mesure la capacité du modèle à identifier correctement toutes les instances positives réelles. Il est calculé en divisant le nombre de vrais positifs (TP) par la somme des vrais positifs et des faux négatifs (TP + FN).

  Recall = TP / (TP + FN)
- **Precision (précision) :** Elle mesure la précision du modèle dans la prédiction des instances positives par rapport à toutes les instances prédites comme positives. Elle est calculée en divisant le nombre de vrais positifs (TP) par la somme des vrais positifs et des faux positifs (TP + FP).

  Precision = TP / (TP + FP)
- **F-Measure :** Elle combine le recall et la precision en une seule métrique pour fournir une mesure équilibrée de la performance du modèle. Elle est calculée en prenant la moyenne harmonique du recall et de la precision.

  F-Measure = 2 * (Precision * Recall) / (Precision + Recall)

Ces métriques sont calculées à partir de la 'matrice de confusion'.

La figure 21, représente la structure de base de la matrice de confusion :

	1 (Predicted)	0 (Predicted)
1 (Actual)	True Positive	False Negative
0 (Actual)	False Positive	True Negative

Figure 21 : Le squelette de la matrice de confusion [42]

4.3. Color/Occupancy : Est une métrique utilisée pour évaluer la qualité de la génération de formes en 3D, en prenant en compte à la fois l'occupation des voxels et les couleurs attribuées à ces voxels. L'occupation fait référence à la présence ou à l'absence d'un objet dans un voxel spécifique, tandis que la couleur fait référence aux attributs colorimétriques tels que la teinte, la saturation et la luminosité d'un voxel.

La métrique "Color/Occupancy" mesure à quel point les voxels générés par un modèle de deep learning correspondent aux voxels réels de la forme cible, tant du point de vue de l'occupation que de la couleur.

5. Expérimentations :

Dans cette section, nous examinerons les différentes architectures utilisées dans notre étude.

- **CGAN :**

On a utilisé le modèle CGAN avec vecteur de bruit et sans vecteur de bruit pour la tâche de génération de formes 3D à partir des embeddings de descriptions textuelles.

Le générateur utilise à la fois l'embeddings de texte et le vecteur de bruit pour générer des formes 3D dans le cas du CGAN avec noise et utilise seulement le vecteur d'embedding dans le cas de CGAN sans noise.

On a enlevé le vecteur de bruit de notre modèle CGAN dans le but d'augmenté ca performance, car on a supposé que le noise pourrait le disperser lors de son entraînement.

L'utilisation des embeddings de descriptions textuelles permet d'incorporer les informations sémantiques spécifiques aux objets, tandis que l'ajout de vecteurs de bruit introduit une composante aléatoire, favorisant la variété des formes générées. Cette combinaison d'éléments sémantiques et aléatoires contribue à améliorer la qualité et la diversité des formes 3D générées par le modèle.

- Les paramètres utilisés pour l'entraînement du modèle CGAN avec noise sont les suivants :
  - Nombre d'epochs : 400
  - Batch size : 32
  - Nombre de worker = 4
  - Learning rate = 0.0002
  - Latent vector = 384
  - Shape_size = 32
  - Nombre de channels = 4
  - La forme des shape 3D = (Nombre de channels, Shape_size, Shape_size, Shape_size)
- Les paramètres utilisés pour l'entraînement du modèle CGAN sans 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
  - La forme des shape 3D = (Nombre de channels, Shape_size, Shape_size, Shape_size)

- Voici ci-dessous les schémas qui représentent les fonctions de perte lors de l'entraînement des deux modèles précédents sur 200 epochs :

5.1. CGAN avec noise :

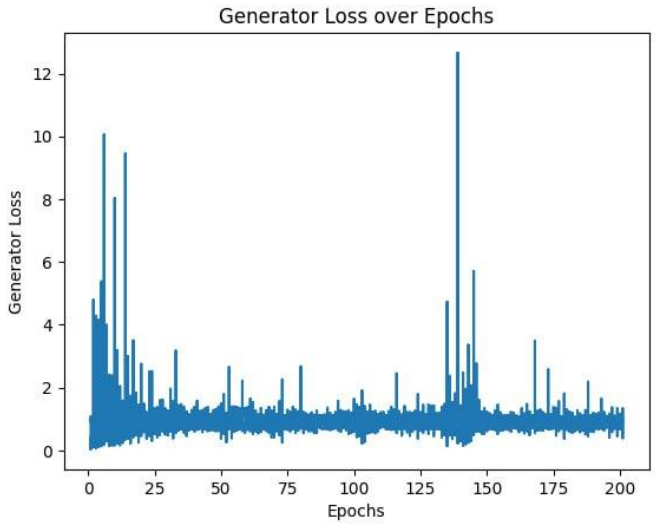

Figure 22 : La perte du générateur en fonction des epochs, CGAN avec noise

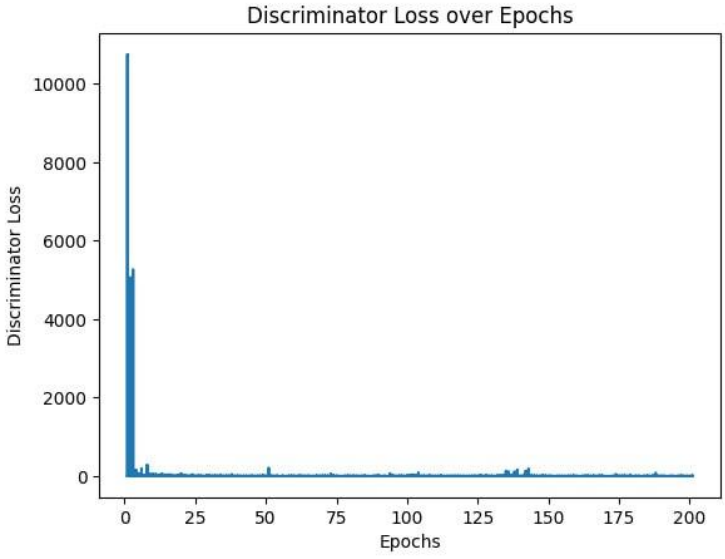

Figure 23 : La perte du discriminateur en fonction des epochs, CGAN avec noise

5.2. CGAN sans noise :

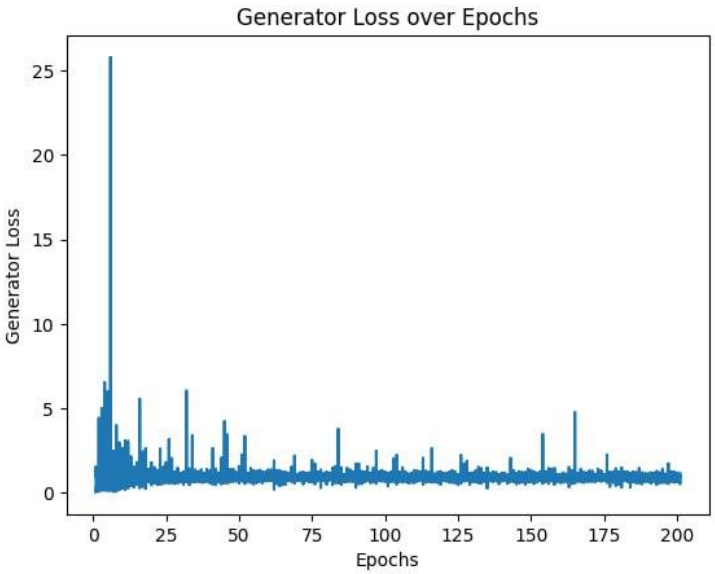

Figure 24 : La perte du générateur en fonction des epochs, CGAN sans noise

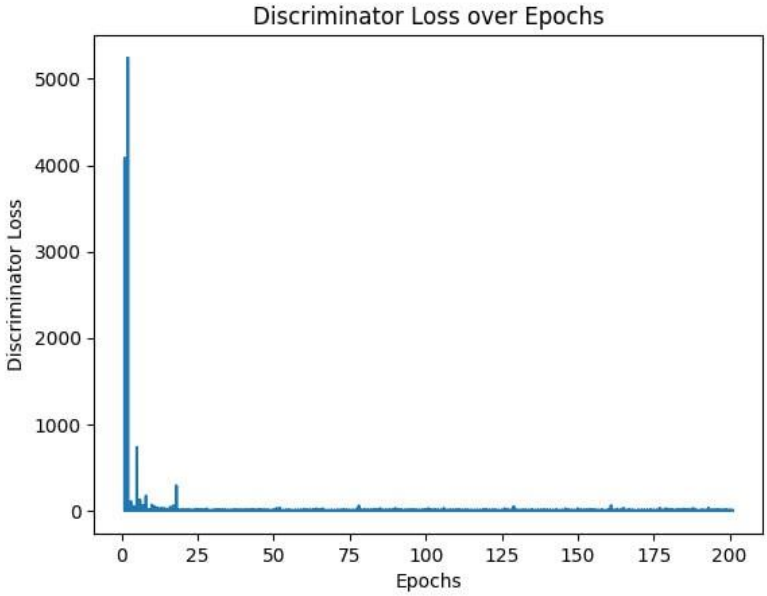

Figure 25 : La perte du discriminateur en fonction des epochs, CGAN sans noise

5.3. Comparaison des pertes des deux modèles :

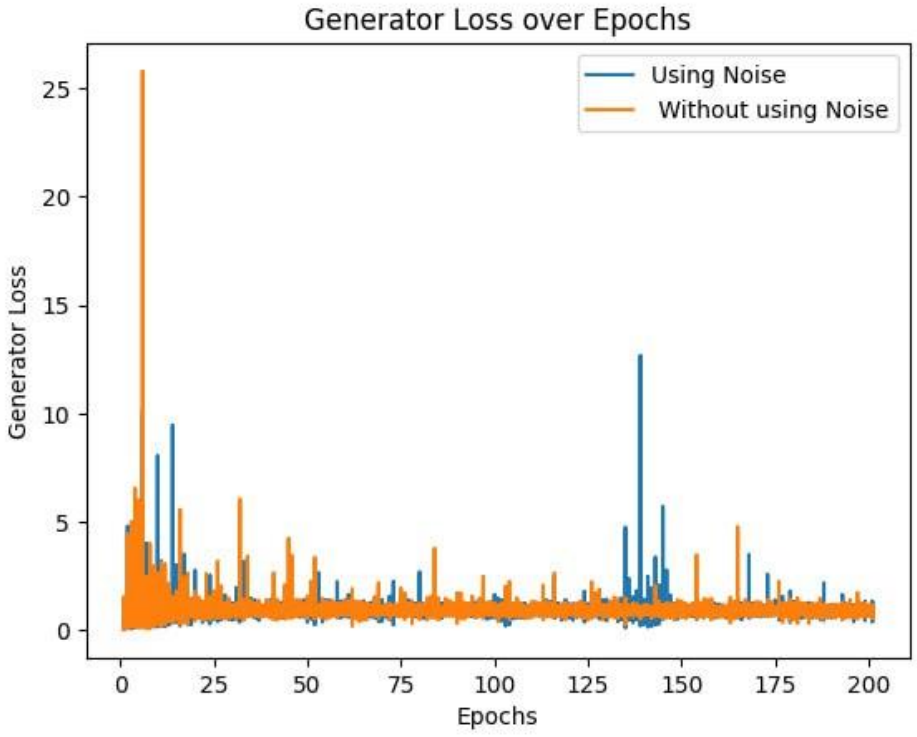

Figure 26 : Comparaison des pertes des deux générateurs, de CGAN avec et sans noise

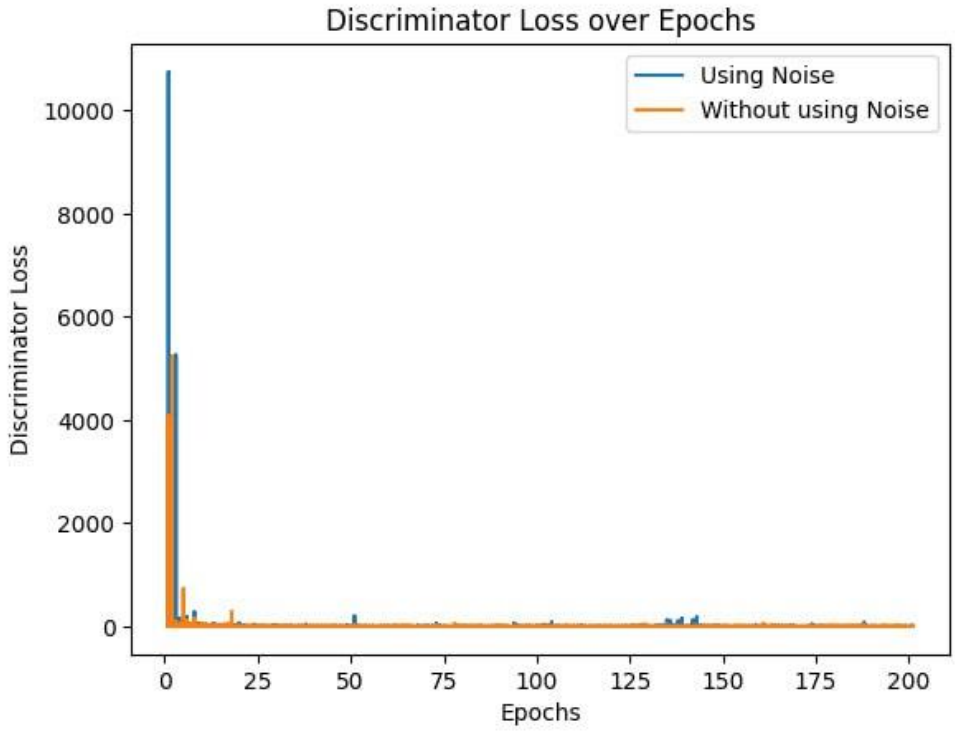

Figure 27 : Comparaison des pertes des deux discriminateurs, de CGAN avec et sans noise

5.4. Discussion :

Dans un CGAN, le générateur et le discriminateur sont en compétition dans un jeu adversarial. Le générateur vise à générer des exemples réalistes pour tromper le discriminateur, tandis que le discriminateur cherche à distinguer les exemples réels des exemples générés.

Idéalement, on souhaite atteindre un équilibre où le générateur est capable de générer des exemples réalistes que le discriminateur a du mal à distinguer des exemples réels. Cela se traduit souvent par une perte faible pour le générateur et une perte faible pour le discriminateur.

Selon les résultats obtenus, on remarque que l'apprentissage du générateur sans vecteur de bruit donne de bonnes résultats pas rapport à l'utilisation du vecteur de bruit et de même pour le discriminateur.

6. Comparaison entre les performances de nos modèles :

Métriques / Modèle	Occupancy			Color/Occupancy
	Recall	Precision	F- Measure	
CGAN Avec Noise	**0.6036**	**0.0700**	**0.1254**	**0.0961**
CGAN Sans Noise	**0.8548**	**0.0701**	**0.1296**	**0.1381**

Tableau 3 : Comparaison entre les performances des modèles utilisés

En utilisant ces résultats d'évaluation, on peut analyser les forces et les faiblesses de nos modèles CGAN.

- CGAN avec noise :

- Notre modèle semble avoir une capacité raisonnable à détecter les formes positives (occupancy), comme le montre le rappel de 0.6036. Cela indique que le modèle parvient à détecter environ 60% des formes positives dans les images générées.

- Cependant, la faible précision de 0.0700 indique que le modèle génère également un grand nombre de faux positifs. Cela signifie que parmi les formes générées, seulement 7% correspondent réellement à des formes positives. Les 93% restants peuvent être des formes incorrectes ou des formes négatives (non occupancy). Autrement dit, les formes générées par

le modèle ne correspondent pas souvent aux attentes ou que le modèle a du mal à générer des formes réalistes.

- La mesure F de 0.1254 suggère un déséquilibre entre la précision et le rappel. Cela est dû à la faible précision du modèle.

- L'occupation de couleur de 0.0961 indique que le modèle génère des images avec une certaine diversité de couleurs qui est relativement faible.

➢ CGAN sans noise :

- Le recall est de 0.8548, ce qui signifie que le modèle a réussi à détecter environ 85,48 % des formes positives présentes dans les données réelles. Un recall élevé indique une bonne capacité du modèle à trouver les formes positives.
- La valeur de 0.0701 pour la précision, indique que seulement environ 7,01 % des formes détectées comme positives par le modèle sont effectivement des formes positives réelles. Cela suggère que le modèle peut générer beaucoup de faux positifs.
- Une F-mesure de 0.1296 indique que notre modèle a une performance relativement faible en termes d'équilibre entre la précision et le recall.
- Une color occupancy de 0.1381 indique que notre modèle génère des données avec une diversité relativement faible en termes de couleurs.

Analyse comparative :

Le modèle 2 (CGAN sans bruit) présente de meilleures performances par rapport au modèle 1 (CGAN avec bruit) en termes de détection des formes positives (occupancy), il a un recall plus élevé, ce qui signifie qu'il est capable de générer un plus grand pourcentage de formes positives dans les données réelles. Les deux modèles ont une précision similaire, ce qui indique qu'ils génèrent un pourcentage similaire de formes positives correctes parmi celles détectées. Cependant, le modèle 2 obtient un meilleur équilibre entre la précision et le recall, mesuré par la F-mesure. De plus, le modèle 2 génère des données avec une plus grande diversité de couleurs par rapport au modèle 1. En conclusion, le modèle 2 est plus performant en termes de recall, F-mesure et diversité des couleurs, mais les deux modèles ont des marges d'amélioration en ce qui concerne la précision.

7. L'Application Desktop qui représente notre projet :

Notre application est une interface graphique développée avec le module 'Tkinter' de Python. Elle permet de convertir une description textuelle en une représentation visuelle 3D en utilisant nos

modèles de génération entrainés. Elle permet aussi d'afficher n'importe quel fichier NRRD contenant une forme 3D stocké dans l'ordinateur.

Notre application possède 6 fenêtres :

- **La fenêtre principale :** Elle est représentée dans la figure 29. C'est La première fenêtre qui s'ouvre l'hors de lancement de l'application. Elle contient deux boutons : Un pour ouvrir la fenêtre de choix de modèle de génération et l'autre pour sélectionner le fichier NRRD qu'on veut l'afficher.

  Pour facilité le lancement de l'application, on a créé un fichier 'Text2Shape.bat' en format batch (fichier de commandes Windows) dans notre application et on a écrit à l'intérieur la commande d'exécution de notre application python. Après en a met un raccourci de ce fichier dans le bureau, comme le montre la figure 28.
- **La fenêtre de choix de modèle :** Elle contient trois boutons pour les trois modèles de génération, comme le montre la figure 30. Chaque bouton ouvre une fenêtre de conversion qui utilise l'un de ces modèles. Elle contient également un bouton 'Retour' pour retourner à la fenêtre principale.
- **Les 2 fenêtres de conversion :** Sont les fenêtres de conversion 'text to shape' où l'utilisateur peut saisir une description textuelle et convertir cette description en un fichier NRRD qui sera met dans le bureau et en une forme visuelle qui sera afficher dans la fenêtre d'affichage, en utilisant le modèle de génération correspondant et la bibliothèque de visualisation 'matplotlib', tout on cliquant sur le bouton 'Convertir'. Elles contiennent aussi un bouton 'Retour' pour retourner à la fenêtre de choix de modèle. La forme de ces fenêtres est présentée par la figure 31.
- **La fenêtre d'affichage :** Qui affiche la forme 3D générée ou le fichier NRRD sélectionné, par 'matplotlib'. Comme le montre les figures 33,34 et 35.

Figure 28 : Un raccourcie pour lancer notre application

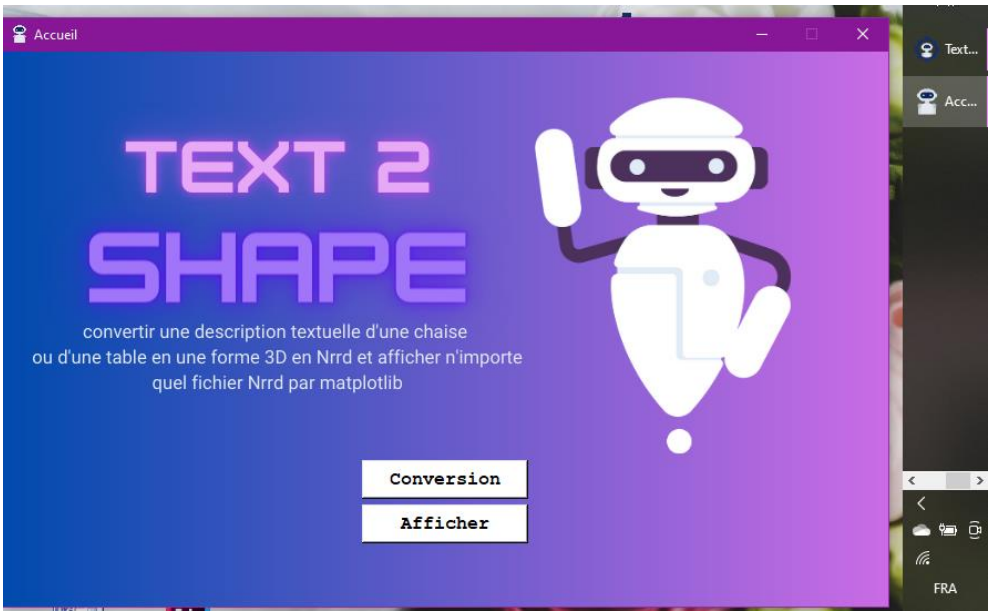

Figure 29 : La fenêtre principale

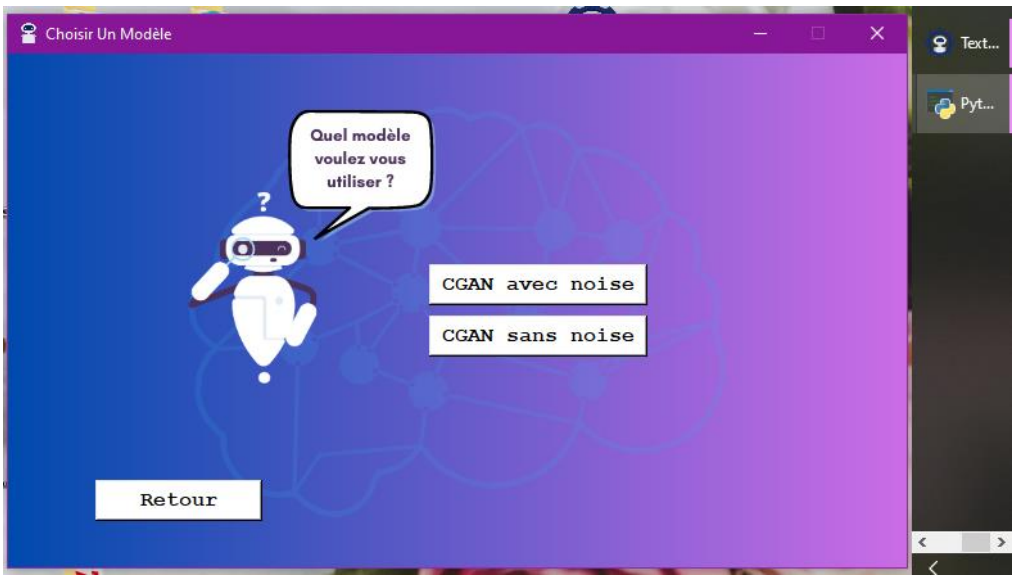

Figure 30 : La fenêtre de choix de modèle

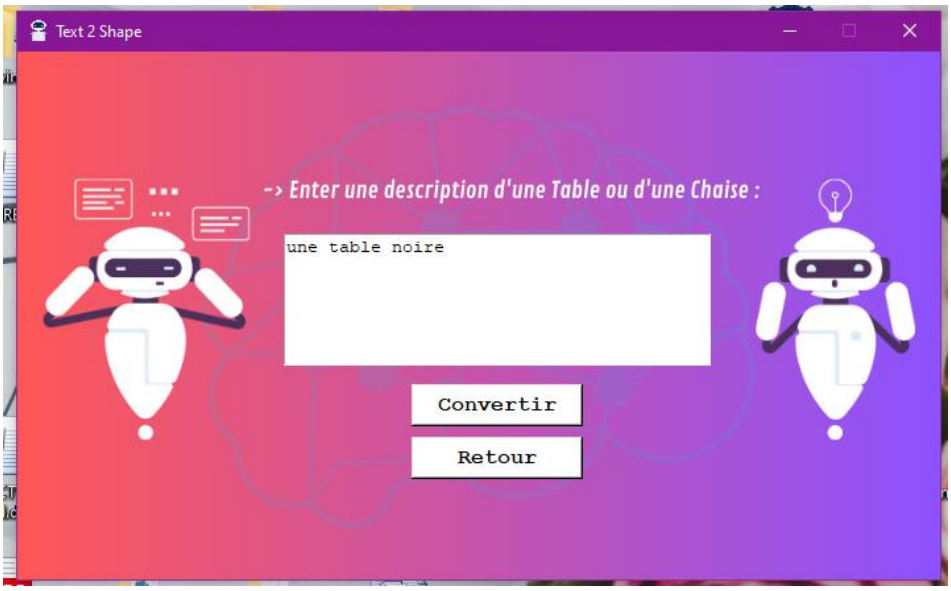

Figure 31 : La fenêtre de conversion

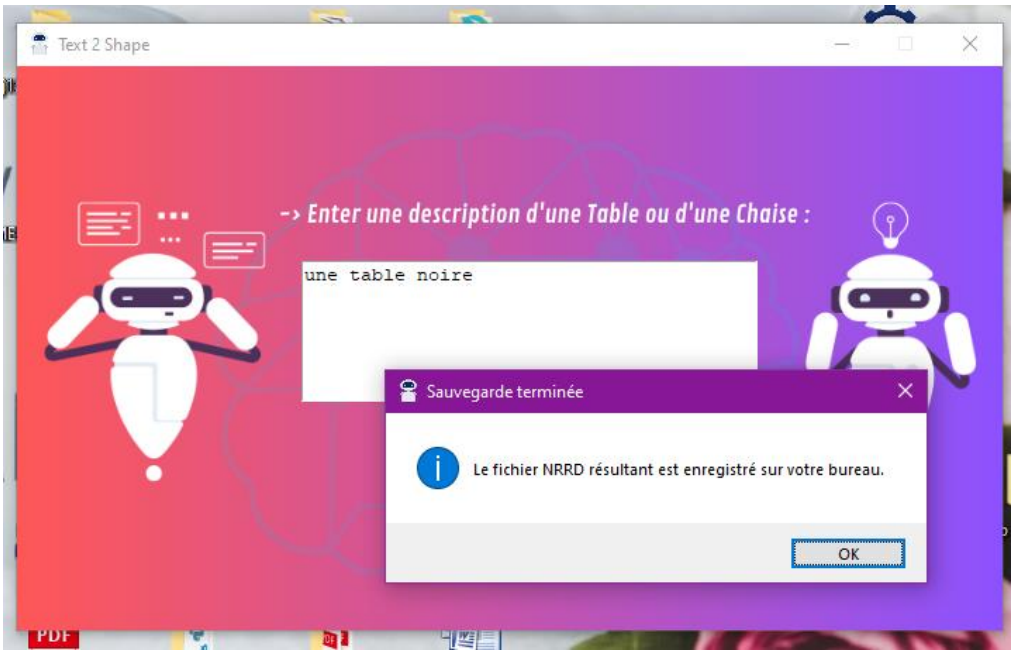

Figure 32 : Une fenêtre de messagerie

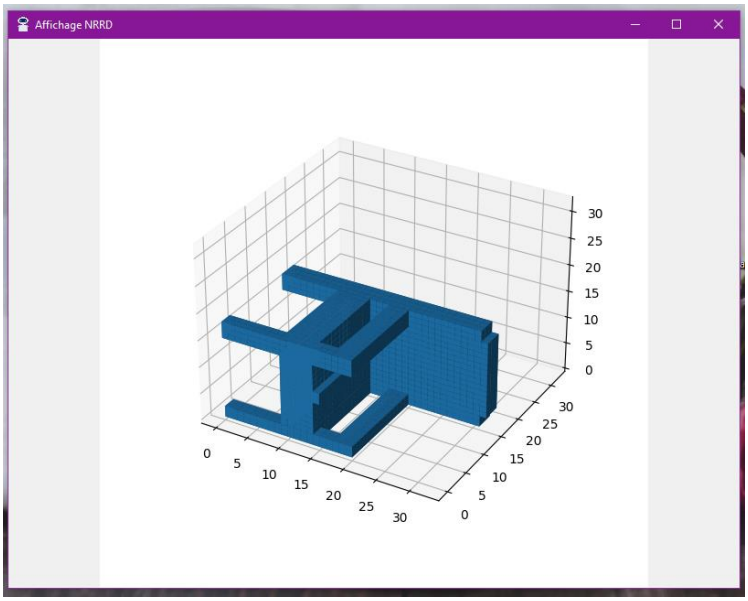

Figure 33 : Affichage d'un fichier NRRD d’une chaise

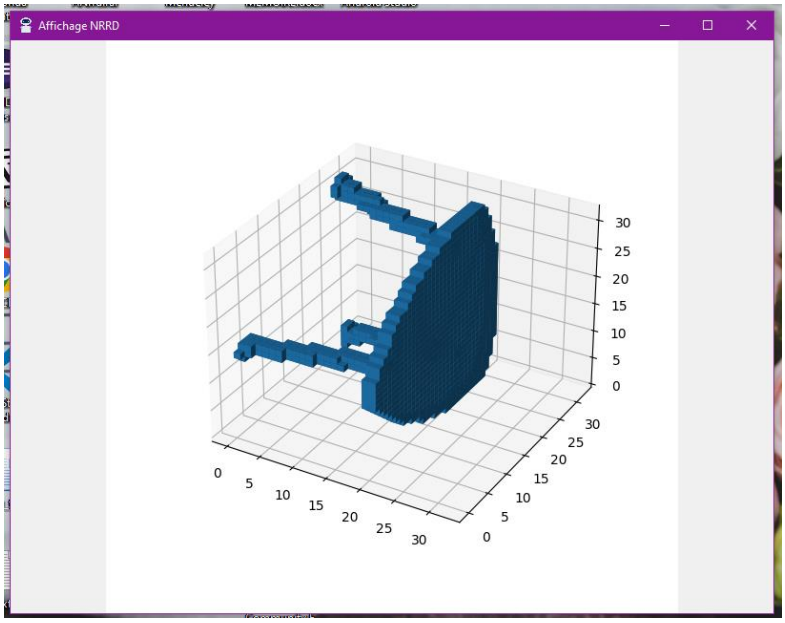

Figure 34 : Affichage d'un fichier NRRD d'une table

8. Discussion :

Nous avons déduit depuis tous les résultats obtenues, que l'apprentissage du générateur sans vecteur de bruit a donné de meilleurs résultats que l'introduction du vecteur de bruit, pas avec une très grande marge, mais qui peut être très utile. Et en ce qui concerne le discriminateur, il performe légèrement mieux sans bruit par rapport à celui avec le bruit.

Donc l'enlève du vecteur de bruit de notre modèle CGAN a amélioré l'apprentissage du générateur et du discriminateur.

Le modèle CGAN sans bruit présentait de meilleures performances en termes de détection des formes positives (occupancy). Il avait un rappel plus élevé, ce qui signifie qu'il était capable de détecter un plus grand pourcentage de formes positives dans les données réelles (85,48%). Mais les deux modèles avaient des précisions similaires faibles (7%), indiquant qu'ils généraient un pourcentage élevé des faux positive. La F-mesure à montré que le modèle sans bruit avait un meilleur équilibre entre la précision et le rappel mais qu'il est faible (12%). Cela suggère que les modèles ont une performance faible en termes d'équilibre entre la détection des formes positives et la réduction des faux positifs.

En ce qui concerne la diversité des couleurs générées, le modèle sans bruit était également meilleur, avec une plus grande diversité par rapport au modèle avec bruit, mais également faible (13%).

- Le modèle CGAN sans noise est plus performant que le CGAN avec noise, avec une amélioration considérablement utile. Il a une bonne capacité de génération mais une faible capacité de prédiction qui peut être évoluée.
- L'enlève du bruit est une bonne décision, pour augmenté les performances de notre modèle CGAN.
- Ces résultats justifient la génération continue d'un cube 3D dans le test de nos deux modèles à travers notre application.

9. Conclusion :

Ce chapitre a permis de mettre en évidence les outils utilisés tout au long de notre projet, le dataset exploité et les résultats obtenus avec une évaluation.

Conclusion Générale

Pour conclure notre mémoire, il est primordial de signaler que la tâche de génération de formes 3D à partir de descriptions textuelles est un domaine de recherche prometteur qui continue de susciter un intérêt croissant de nos jours. Dans le contexte actuel, plusieurs avancées et approches ont été développées pour aborder cette problématique. Cependant, malgré les progrès réalisés, la génération de formes 3D à partir de descriptions textuelles reste un défi complexe à cause de certaines difficultés, telles que la représentation sémantique, la diversité et la qualité des résultats, la cohérence spatiale, etc.

Nous avons participé à cette voie à travers notre travail qu'il est basé sur les formes 3D (chaises et tables) du dataset ShapNet avec leurs descriptions textuelles. On a d'abord convertie les descriptions en vecteurs numériques pour les utilisées comme entrées de nos modèles de génération, pour cela on a utilisé le modèle Transformer pré-entrainé 'all-MiniLM-L6-v2'. Après, on a conçu une architecture CGAN, avec vecteur d'embedding de texte et un vecteur de bruit aléatoire en entrée dans la première expérimentation, et avec seulement le vecteur d'embedding de texte dans la deuxième expérimentation dans le but d'augmenter la performance de notre modèle. Le développement s'est fait dans l'environnement Google colab. Et finalement, en a développé une interface graphique avec Python, pour tester nos modèles.

On a pu développer un modèle CGAN (sans noise) performant en génération de forme 3D mais qu'il est faible en prédiction, qui peut être améliorée en prenant en considération les propositions suivantes : Il est vivement recommandé d'utiliser une autre méthode pour la génération des text embeddings, plus pertinente et plus performante pour améliorer les résultats du modèle. Il est possible d'essayer d'autres valeurs de paramètres de modèle, de tester notre approche sur un autre dataset que ShapeNet ou d'utiliser d'autres types de formes 3D. Il est aussi possible d'utiliser d'autres architectures des réseaux antagonistes génératifs ou d'autres types de modèles génératifs.

En conclusion, notre travail est une participation au défi de l'automatisation de la tache 'text to shape', qu'elle vise à fournir une solution efficace et performante pour les concepteurs et designers de formes 3D pour économiser leur temps, améliorer leurs résultats et augmenter leur productivité.

Références bibliographiques

[1] “Machine learning.” https://www.intelligence-artificielle-school.com/machine-learning-quest-ce-que-cest/

[2] Shalev-Shwartz, S. & Ben-David, S. (2014). "Understanding Machine Learning:From Theory to Algorithms". Cambridge University Press

[3] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of StatisticalLearning: Data Mining, Inference, and Prediction (2nd ed.). Springer.

[4] “machinelearnia”, [Online]. Available: https://machinelearnia.com/apprentissage-supervise-4-etapes/

[5] [Online]. Available: https://alioh.github.io/Machine-Learning-for-Everyone-4/

[6] (incompleteideas.net, 2020). Reinforcement Learning: An Introduction

[7] [Online]. Available: https://simseo.fr/blog/page/126/

[8] “François Chollet - Deep Learning with Python-Manning (2017).”

[9] “(Adaptive Computation and Machine Learning series) Ian Goodfellow, Yoshua Bengio, Aaron Courville - Deep Learning-The MIT Press (2016).pdf.”

[10] Nielsen, M. A. (2015). "Neural Networks and Deep Learning: A Textbook".Determination press.

[11] A. Chahmi, “Identification paramétrique de la machine asynchrone dédiée au diagnostic Abdelghani Chahmi To cite this version : HAL Id : tel-01658902 Présentée par : CHAHMI Abdelghani Intitulé Identification paramétrique de la machine asynchrone dédiée au diagnostic,” 2017.

[12] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[13] “Réseaux convolutifs”, [Online]. Available: https://blent.ai/blog/a/cnn-comment-ca-marche

[14] H. Gholamalinezhad and H. Khosravi, “Pooling Methods in Deep Neural Networks, a Review,” 2020, [Online]. Available: http://arxiv.org/abs/2009.07485

[15] “Classifying images of Cats and Dogs using Convolutional Neural Networks (CNNs)”, [Online]. Available: https://larbifarihi.medium.com/classifying-cats-and-dogs-using-convolutional-neural-networks-cnns-ce6bc7bad64d

[16] l'article original de Ian Goodfellow et al., intitulé "Generative Adversarial Networks" publiéen 2014.

[17] S. Mirza, M., & Osindero, “Conditionalgenerativeadversarial nets. arXivpreprint arXiv:1411.1784,” 2014, [Online]. Available: https://arxiv.org/abs/1411.1784

[18] [Online]. Available: https://seunghan96.github.io/dl/gan/CGAN_pytorch/

[19] [Online]. Available: https://blog.negativemind.com/2019/10/05/conditional-gan/

[20] A. C. et P. V. Yoshua Bengio, “Representation learning: A review and new perspectives,” *Rev. IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, pp. 1798–1828, 2013.

[21] [Online]. Available: https://lilianweng.github.io/posts/2018-08-12-vae/

[22] A. Veltman, D. W. J. Pulle, and R. W. De Doncker, “The Transformer,” *Power Syst.*, no. Nips, pp. 47–82, 2016, doi: 10.1007/978-3-319-29409-4_3.

[23] K. Crane, “Discrete Differential Geometry: an Applied Introduction,” *Dict. Genomics, Transcr. Proteomics*, 2020.

[24] D. J. & J. H. Martin., “Speech and Language Processing: An introduction to natural language processing,” *SPEECH Lang. Process. An Introd. to Nat. Lang. Process. Comput. Linguist. Speech Recognit.*, pp. 1–18, 2001, [Online]. Available: http://www.cs.colorado.edu/~martin/slp.html

[25] “Pre - Training and Fine - Tuning BERT for the,” 2023.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc.*, pp. 1–12, 2013.

[27] K. Jing and J. Xu, “A Survey on Neural Network Language Models,” 2019, [Online].

Available: http://arxiv.org/abs/1906.03591

[28] Mortenson, M. E. (2018). Mathematics for Computer Graphics and Game Programming: ASelf-Teaching Introduction with CD-ROM. Cengage Learning

[29] C. R. Qi, H. Su, M. Niebner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 5648–5656, 2016, doi: 10.1109/CVPR.2016.609.

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 77–85, 2017, doi: 10.1109/CVPR.2017.16.

[31] Z. Wu *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1912–1920, 2015, doi: 10.1109/CVPR.2015.7298801.

[32] Tiantian Wang, Xiaohu Guo, and Ruigang Yang. 2020. Generating 3D Shapes fromText Using Wasserstein GANs. ACM Trans. Graph. 39, 4, Article 110 (July 2020), 12 pages.DOI:https://doi.org/10.1145/3386569.3392382

[33] B. Hao, Z., Mo, Q., Zhuang, Y., Cao, Y., & Chen, "Joint Word-Shape Embedding for Generating 3D Shapes from Text Descriptions.," vol. 43, pp. 3161-3176., 2021, doi: 10.1109/TPAMI.2020.3010875.

[34] Xinyi Yang, Peng Song, Chi Zhang, and Hao Zhang. Adversarial Synthesis of 3DObject Shape and Appearance from Sketches and Text. ACM Transactions on Graphics (TOG),39(4):Article 42, July 2020.

[35] Xiaoyu Liu, Yiqun Lin, Yijuan Lu, Ming Tang, Jingyi Yu, and Hui Huang. Shapeand Texture Generating Multi-Modal GAN for Joint Attributes and Object Stages Editing. IEEETransactions on Visualization and Computer Graphics, vol. 27, no. 4, pp. 1694-1704, Apr. 2021

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

[37] http://www.mickaeltits.be/super_resolution/

[38] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional

transformers for language understanding," *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

[39] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," *EMNLP-IJCNLP 2019 - 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf.*, pp. 3982–3992, 2019, doi: 10.18653/v1/d19-1410.

[40] C. Bishop, "Pattern Recognition and Machine Learning".

[41] "Text2Shape." http://text2shape.stanford.edu

[42] "Tools for Machine Learning Performance.", [Online]. Available: http://aimotion.blogspot.com/2010/09/tools-for-machine-learning-performance.html