REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB BLIDA

FACULTE DES SCIENCES

DEPARTEMENT INFORMATIQUE

Mémoire De Fin D'études

Pour L'obtention Du Diplôme de master en Informatique

OPTION : Ingénierie des logiciels

Présenté Par :

BOUCHAMA Naziha

THEME

Génération des formes 3D à partir des descriptions textuelles

Soutenu Publiquement le 27 /09/2022

Devant le jury composé de :

Encadreur:

Mr A.H KAMECH

Président (e) :

Mme I. CHERFA

Examinateur :

Mr S. BENAISSI

Année universitaire

2021/2022

Résumé :

Actuellement, avec les nouvelles applications de la 3D notamment dans la réalité virtuelle, la conception de systèmes 3D est de plus en plus demandée.

Toutefois, cette tache nécessite beaucoup de travail et de couts pour les designers, afin de faciliter ce travail, on s'intéresse dans notre projet à la génération des formes 3D à partir des descriptions textuelles, en se servant de la puissance de quelques variantes des réseaux de neurones antagonistes génératifs (CGAN, WGAN), la méthode BERT pour la manipulation des entrées textuelles tout en réduisant le volume des formes 3D à l'aide des autoencodeurs.

Nos modèles ont été testés sur le dataset ShapeNet.

Mots clés : BERT, génération des formes 3D, réseaux de neurones, CGAN, WGAN, autoencodeurs.

ABSTRACT:

Currently, the new applications of 3D, particularly in virtual reality, the design of 3D systems is increasingly in demand.

However, this task requires a lot of work and costs for designers, in order to facilitate this work, we are interested in our project in the generation of 3D shapes from textual descriptions, using the power of some variants of Generative Adversarial Neural Networks (CGAN, WGAN), BERT method for manipulating text inputs and reducing the volume of 3D shapes using autoencoders.

Our models have been tested on the ShapeNet dataset.

Key Words: BERT, 3D shape generation, neural networks, CGAN, WGAN, autoencoders.

ملخص

حاليًا، ظهور التطبيقات الجديدة ثلاثية الأبعاد خاصة في ميدان الواقع الافتراضي حيث يزداد الطلب على تصميم هاته الأنظمة يوما بعد يوم.

ونظرا لحاجتها الكثيرة من العمل والتكاليف للمصممين و من أجل تسهيل العمل، اهتممنا بمشروعنا الحالي في إنشاء أشكال ثلاثية الأبعاد من أوصاف نصية، باستخدام قوة بعض المتغيرات من الشبكات العصبية الانشائية ((WGAN ،CGAN) ، و باستعمال طريقة BERT لمعالجة مدخلات النص مع تقليل حجم الأشكال ثلاثية الأبعاد باستخدام أجهزة التشفير التلقائي.

تم اختبار النماذج الخاصة بنا على مجموعة بيانات ShapeNet.

كلمات مفاتيح : BERT إنشاء أشكال ثلاثية الأبعاد, الشبكات العصبية,CGAN, WGAN , التشفير التلقائي.

Remerciements

Je remercie le bon dieu miséricordieux de m'avoir donné la santé, la chance, la force et la patience afin de terminer.

Un grand merci à l'encadreur **Mr KAMECH** tout d'abords d'avoir accepté de m'encadrer, de m'avoir également permis à travers ce projet d'approfondir mes connaissances et d'apprendre encore plus sur le domaine de l'apprentissage et aussi pour ses précieux conseils.

J'exprime ma gratitude envers tous les enseignants du département informatique de Blida,

Mes remercîments les plus distingués aux membres du jury **Mme CHERFA** et **Mr BENAISSI** d'avoir accepté d'examiner mon travail.

Je remercie toute les personnes qui m'ont soutenu et ayant participé de près ou de loin à la concrétisation de ce mémoire.

Naziha

Dédicace

Je dédie ce travail à mon défunt père qui attendait impatiemment la finalisation de mon parcours mais qui est parti très tôt.

À ma très chère maman, qui m'a encouragé et soutenu tout au long de mon parcours et qui m'a toujours aidé à me relever.

à mes frères RAOUF, CHAHER

À ma meilleur amie INES qui m'encourageait constamment durant mes périodes les plus difficiles ….

Naziha

TABLE DES MATIÈRES

Liste des tableaux

Liste des figures

Liste des abréviations

Abréviation	Description complète
GAN	Generative Adversial Network
CGAN	Conditional Generative Adversial Network
WGAN	Wasserstein Generative Adversial network
BERT	Bidirectional Encoder Representation from Transformers
LSTM	Long Short Term Memory
RNN	Recurrent neural Network
ReLU	Rectified Linea Unit

Introduction Générale

Introduction générale

Depuis quelques années, la combinaison entre l'intelligence artificielle et la visualisation 3D permet de répondre aux exigences de divers secteurs industriels. De grandes entreprise comme Facebook se sont tournées vers le domaine de la 3D afin de fournir aux utilisateurs un contenu riche et de meilleur performance pour une réalité augmentée.

Avant, les concepteurs avaient tendance à se servir des applications de modélisation 3D (Maya, Blender …) pour accomplir leurs travaux mais ça s'avère très couteux, c'est pourquoi quelques chercheurs se sont focalisés sur l'étude des différentes approches de génération 3D dont est le but de notre projet.

L'approche de génération des formes 3D peut aider les architectes et les designers comme par exemple le design des structures et l'aménagement intérieurs des maisons et peut aussi être utile pour les créateurs des jeux vidéo. Toutefois, les formes 3D consomment beaucoup d'espace de stockage.

L'objectif majeur de notre travail est de concevoir un modèle qui pourra comprendre les mots des descriptions textuelles dans le but de générer la forme 3D correspondante dans un espace réduit en se servant à la fois la puissance des architectures antagonistes, des autoencodeurs pour traiter les formes 3D et le modèle BERT pour la transformation des descriptions textuelles.

Les principales tâches sont :

- Construire un modèle de réseaux de neurones pour apprendre des représentations des formes 3D.
- Construire un modèle pour associer les formes 3D et leurs descriptions textuelles
- Valider le modèle en utilisant le dataset Shapenet.

Notre mémoire est divisé en deux parties principales : la partie théorique est composé des deux premiers chapitres et la partie conception réalisation pour le reste.

- **Chapitre 1 :** ce chapitre est consacré à l'apprentissage automatique et les réseaux de neurones ou nous expliquerons ses différentes architectures.
- **Chapitre 2 :** dans ce chapitre nous présenterons les travaux connexes en relation avec l'approche text to shape.
- **Chapitre 3 :** on va détailler les méthodes utilisées ainsi que l'architecture de notre approche.
- **Chapitre 4 :** dans ce dernier chapitre on va présenter les résultats obtenus ainsi que les outils et l'environnement utilisé.

Chapitre 1 : Apprentissage automatique

1. Introduction

L'apprentissage automatique est une branche de l'intelligence artificielle destiné à imiter l'intelligence humaine, son objectif est de développer de nouveaux modèles en utilisant les différents algorithmes et outils qui trouve leurs applications dans diffèrent domaines notamment la vision par ordinateur, les finance, traitement du langage naturel …etc

Dans ce chapitre nous ferons une représentation de l'apprentissage automatique et ses différents outils et nous expliquerons l'apprentissage profond et ses différentes architectures.

2. L'apprentissage automatique

Selon Arthur Samuel , fondateur de ce domaine , le Machine Learning consiste à laisser l'ordinateur apprendre quel calcul effectuer, plutôt que de lui donner ce calcul (c'est-à-dire le programmer de façon explicite) [1].

L'apprentissage automatique permet d'analyser et de modéliser selon un ensemble d'exemples afin de produire des sorties selon l'objectif définit par l'expert.

Parmi les méthodes d'apprentissage les plus utilisé on retrouve :

- L'apprentissage supervisé.
- L'apprentissage non supervisé.
- L'apprentissage par renforcement.

2.1. L'apprentissage supervisé

Dans cette approche, l'ensemble des données en entrées possèdent une étiquette au préalable, c'est-à-dire que chaque exemple a une sortie qui est connu à l'avance, deux types d'algorithmes d'apprentissage supervisé diffèrent [2]:

La Régression : dans la mesure où la variable de sortie est numérique comme par ex prédire le prix d'un appartement…

La Classification : utilisé dans le cas où la variable de sortie est purement qualitative comme par exemple classifier [chat, chien].

L'apprentissage supervisé utilise un ensemble d'algorithmes, les plus utilisés sont : les arbres de décision, KNN, régression linéaire….

La figure suivante montre un exemple de classification de spam :

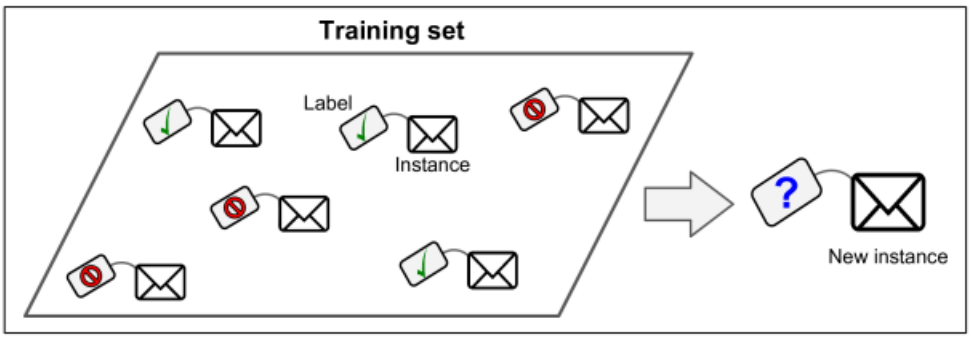


Figure 1: apprentissage supervisé cas de classification de spam [3]

2.2. L'apprentissage non supervisé

Contrairement à l'apprentissage supervisé, les exemples d'entrées ne sont pas étiquetés, son objectif sera de modéliser la structure ou la distribution sous-jacente dans les données afin d'en apprendre d'avantages , on retrouve deux catégories [4] :

- *Le regroupement* :il s'agit de regrouper les paramètres des objets selon leurs similarités dans des clusters l'algorithme le plus utilisé est KMeans.
- *L'association* : consiste à lier les données conformément à un ensemble de caractéristiques comme le cas de la problématique du panier de la ménagère, l'algorithme le plus utilisé est les règles d'association.

La figure suivante montre un exemple de regroupement (clustering) :

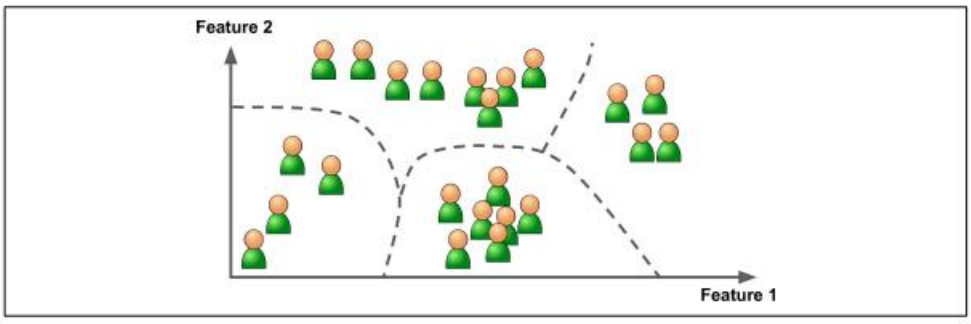


Figure 2: clustering [3]

2.3. Apprentissage par renforcement

Dans ce type d'apprentissage, l'agent est une entité qui interagit avec son environnement et réalise un ensemble d'actions, semblable à l'apprentissage d'un humain avec son environnement, il observe les résultats de ses interactions, si c'est une bonne action l'agent recevra une récompense dans le cas contraire il recevra une pénalité, l'idée est que l'agent doit maximiser ses performances et d'apprendre uniquement par son expérience [5] ,comme illustré dans la figure suivante :

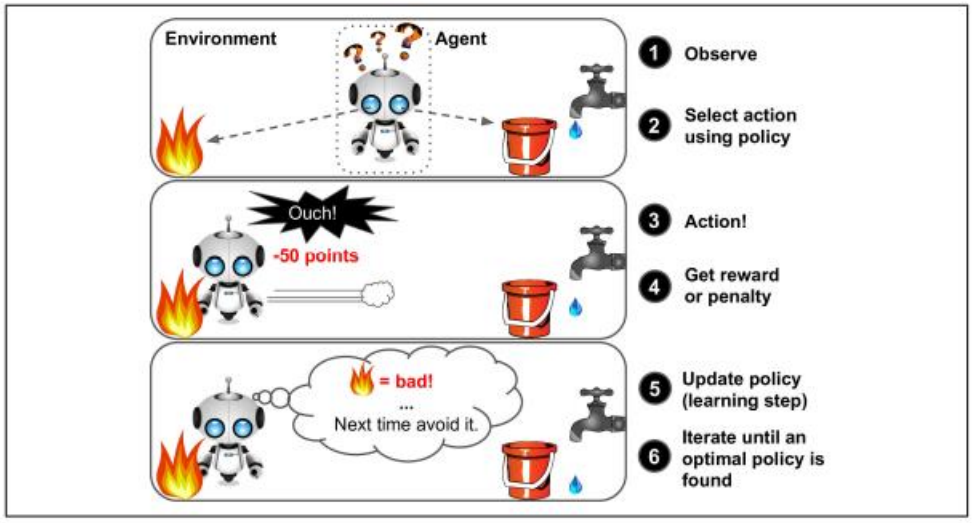


Figure 3 : apprentissage par renforcement [3]

3. Apprentissage profond

C'est un sous domaine de l'apprentissage automatique qui apparait dès les années 2010, son architecture de base est constituée d'un ensemble de neurones interconnecté avec d'autres neurones pour former un réseau et empilé en un ensemble de couches [6].

À la base, il existe trois couches : une couche d'entrée, une couche cachée et une couche de sortie, plus le nombre de couche caché est profond, plus le réseau sera capable de retrouver des solutions à des problèmes complexes, comme le montre la figure suivante :

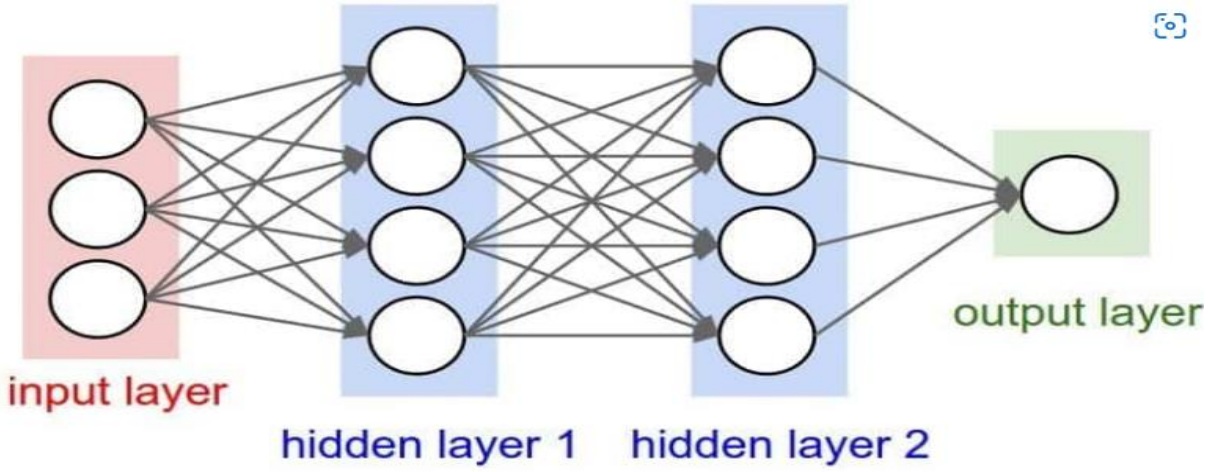


Figure 4: architecture de base d'un réseau de neurones [7]

3.1. Neurone biologique

Le système nerveux du cerveau se compose principalement d'une centaine de milliards de neurones , chaque neurone est composé d'un corps cellulaire **soma** qui se ramifie pour former des **dendrites** (récepteurs du signal), le signal sera prolongé le long d'un **axone** et le **synapse** à l'extrémité de l'axone qui est la sortie vers les autres voisins [8] , illustré dans la figure suivante :

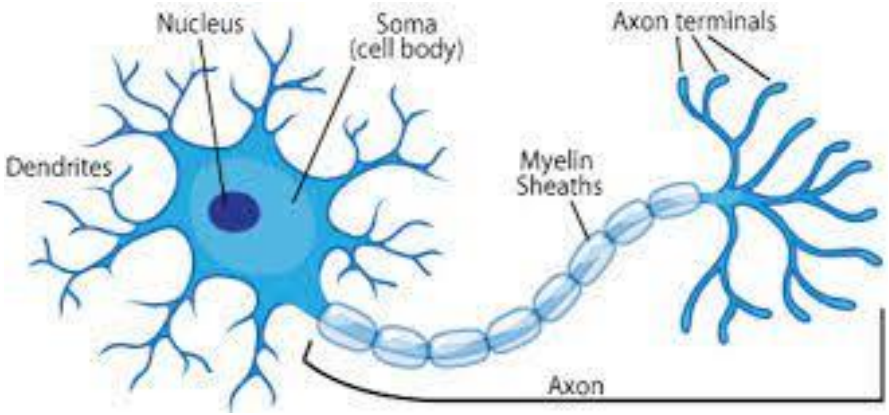


Figure 5: neurone biologique [9]

3.2. Neurone formel

Le neurone représente un processeur élémentaire créé par McCulloh [10] , inspiré essentiellement du neurone biologique ,chaque neurone possède un ensemble d'entrée x1….xn, chaque entrée est muni d'un poids **w1…wn** , une fonction d'activation **ϕ** et une sortie **y,** comme le montre la figure suivante:

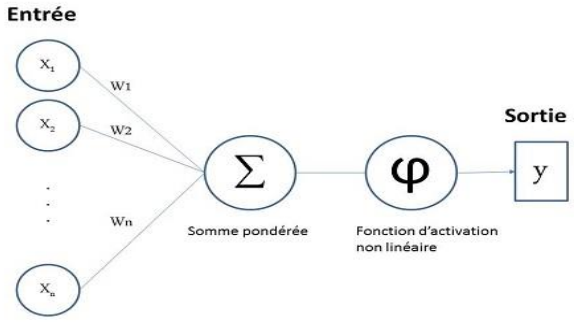


Figure 6: modèle neurone formel [11]

3.3. La fonction d'activation

La fonction d'activation permet de calculer la sortie du neurone en utilisant la somme pondéré des entrées [12] , en utilisant l'équation suivante :

$$sortie = \phi\ (\sum(xi * wi))\ i = 1 \dots \dots n$$

xi : le vecteur d'entrée

wi: représente le poids

ϕ: la fonction d'activation

Sortie : représente le résultat

Il existe un ensemble de fonctions d'activation qui sont utilisés pour atteindre la sortie désirée, dans le cas d'une fonction linéaire, la sortie ne sera limitée à aucune plage, ce qui produit un signal de sortie proportionnelle à l'entrée ; très peu utilisé, les chercheurs ont recours à la fonction non linéaire qui est mieux adapté et qui peut prendre différentes formes suivant l'application le tableau suivant montre les fonctions d'activation les plus utilisés :

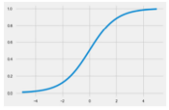
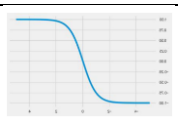
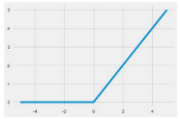
Fonction	Equation	Schéma
Sigmoïde	$\frac{1}{1 + e - x}$	
TanH	$\frac{e\,x - e - x}{e\,x + e - x}$	
RELU	$\max(0, x)$	

Tableau 1 : les différentes fonctions d'activation [13]

3.4. La fonction de perte

Lors de l'entrainement du dataset , pour chaque entrée, il est nécessaire de calculer le taux d'erreur qui représente la sortie comparé à la fonction de sortie désirée , ce processus se fait pour chaque epoch et les transmettra à une fonction de perte [14].

3.5. La descente du gradient

C'est un algorithme d'optimisation utilisé pour minimiser la fonction de cout, le principe est de calculer la *dérivée partielle* de chaque paramètre (poids) [13],il existe trois types : la descente de gradient par lot , descente de gradient mini lot et la descente de gradient stochastique.

La descente de gradient par lot : ce type d'algorithme traite tous l'ensemble de donnée d'apprentissage, sauf qu'elle est très couteuse si le nombre est important et il est déconseillé de l'utiliser.

La descente de gradient mini lot : la descente de gradient par mini-lots l'algorithme calcule le gradient sur de petits sous-ensembles d'observations sélectionnées aléatoirement qu'on appelle mini-lots [15]. Ce qui améliore les performances.

La descente de gradient stochastique :dans ce type d'algorithme il faut choisir à chaque étape une observation prise au hasard dans l'ensemble d'entraînement et

calcule les gradients en se basant uniquement sur cette seule observation [15]. Ce qui rend l'algorithme beaucoup plus rapide puisqu'il n'a que très peu de données à manipuler à chaque itération et pourra aussi entrainer un ensemble de donnée très grand.

4. Les architectures du Deep–learning

Il existe une multitude d'architectures de réseaux de neurones, nous citons dans ce qui suit les plus répandus :

4.1. Les réseaux de neurones convolutifs

Ce type d'architecture est utilisé principalement dans la classification d'image, il est composé de plusieurs couches :

4.1.1. La couche de convolution

L'idée principale de la convolution est d'utiliser une matrice appelé filtre sur l'ensemble de l'image en entrée et en déduire une autre image [16].

Principalement le filtre sera placé de haut à gauche de l'image à classifier, la valeur du pixel de l'image sera multipliée par celle du filtre et la somme de ces produits sera ainsi réalisée, le filtre sera ensuite déplacé selon le nombre de stride, si le stride =1 le déplacement se fait d'1 ligne et d'1 colonne ainsi de suite jusqu'à le filtrage de l'ensemble de l'image comme l'illustre la figure suivante :

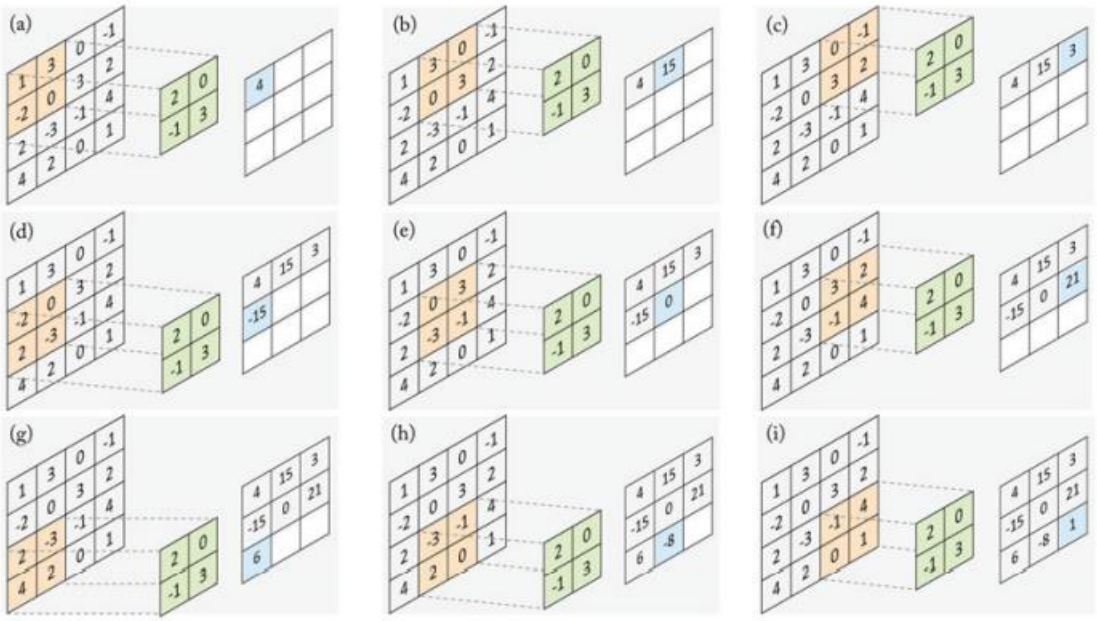


Figure 7: Le fonctionnement d'une couche de convolution [17]

4.1.2. La couche pooling

L'image issue de la couche de convolution lui sera appliqué un ensemble de traitements le pooling, son principal objectif est de réduire la dimension de l'image, tout comme l'opération de convolution, la taille du noyau sera prédéfinie et le stride aussi, il existe trois types :

- *Max Pooling, :* il s'agit de rechercher la valeur maximale de la fenêtre,
- *Average Pooling :* permet de calculer la moyenne de chaque fenêtre sera calculé.
- *Pooling stochastique* dans ce type de pooling , une valeur sera retenue selon une estimation probabilistique [18] .

La figure suivante montre les étapes de *max pooling* pour une région de dimension 2*2 et un stride =1:

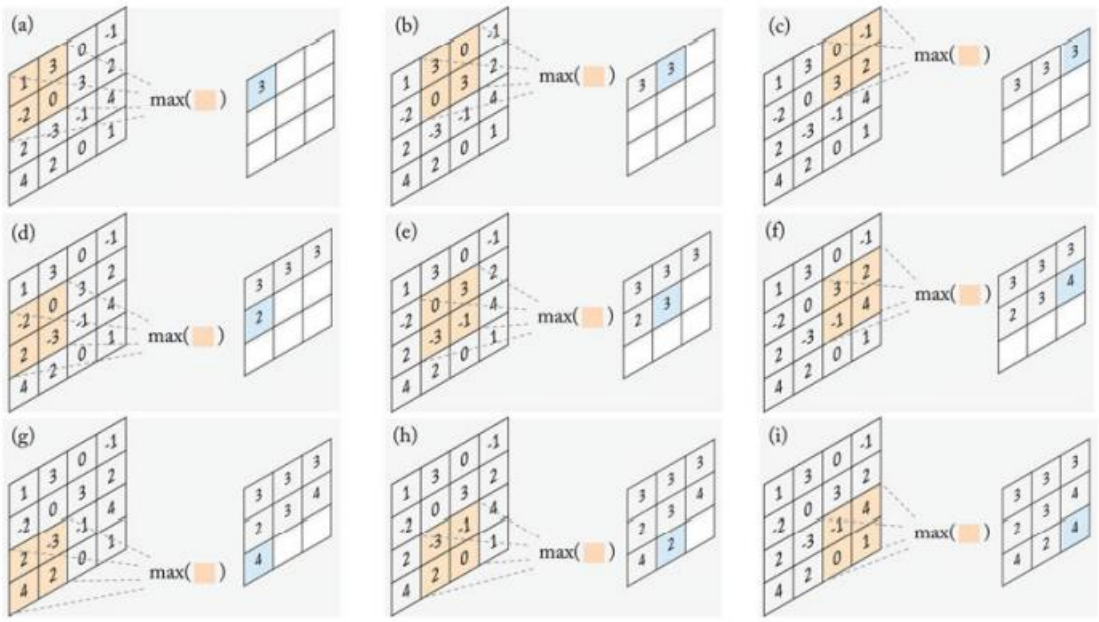


Figure 8: l'opération du Max Pooling [17]

4.1.3. La couche entièrement connecté (fully connected layer)

Elle représente la dernière couche de l'architecture CNN(convolutional neural network), entièrement connectées à tous les neurones de sortie ,après avoir reçu les vecteurs d'entrée, cette couche applique séquentiellement des combinaisons linéaires et une fonction d'activation afin de classer l'image d'entrée , en sortie, un vecteur de taille égal au nombre de classes sera renvoyé où chaque composante représente la probabilité que l'image d'entrée qui appartient à une classe [19], comme illustré dans la figure suivante :

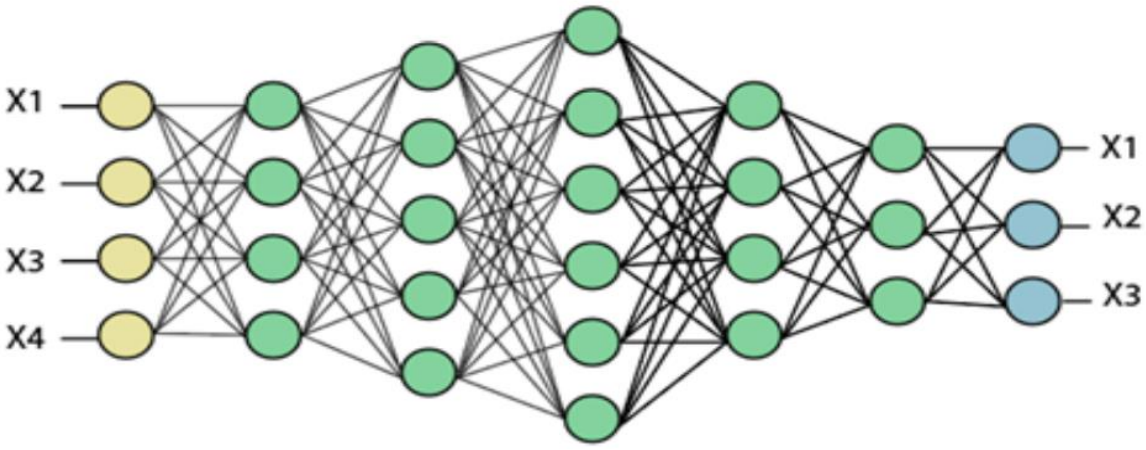


Figure 9: la couche entièrement connecté [20]

4.2. Réseau de neurones récurrents

Contrairement aux réseaux de neurones à convolution utilisé dans le domaine de la classification des images , les réseaux de neurones récurrents sont utilisés dans le domaine de traitement du langage naturel ,à l'opposé des autres architectures, les réseaux de neurones récurrents ont la particularité de faire revenir en arrière certaines connexion ,ce type de modèle souffre de mémoire courte si la séquence est longue [21] , la figure suivante montre l'architecture d'un réseau de neurones récurent :

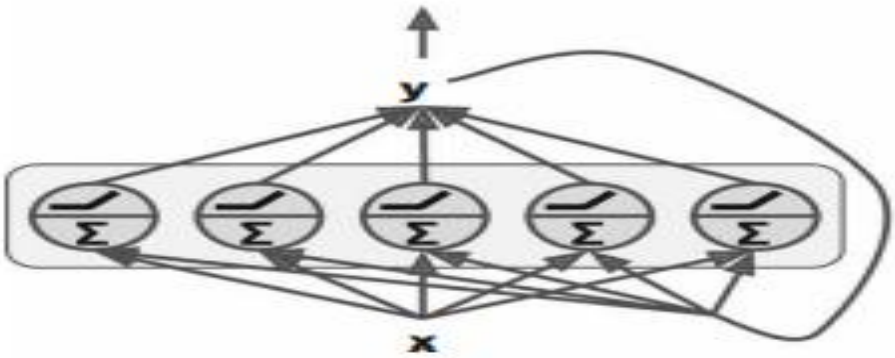


Figure 10: architecture de réseau de neurone récurent [15]

Parmi les applications RNN(recurrent neural network) les plus utilisés nous citons :

- *one to many* :dans ce type d'architecture une seule entrée est reçu dans le RNN et plusieurs sorties sont retournés exemple la légende d'image.
- *many to one* le RNN possède plusieurs entrées avec une seule sortie un exemple de ce mode est l'analyse de sentiment sur des textes.
- *many to many* dans ce type le RNN prend plusieurs entrées et produit plusieurs sorties même si le nombre de neurones d'entrée et de sortie diffèrent, exemple la traduction de texte .

La figure suivante montre les différentes architectures RNN :

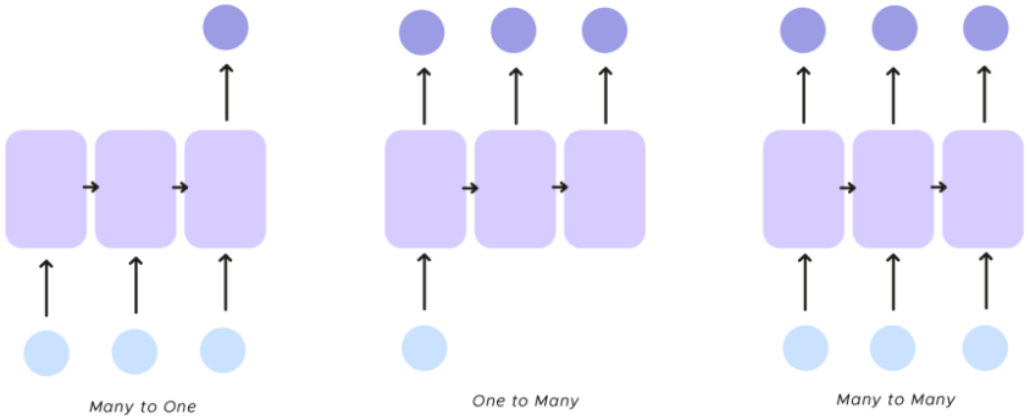


Figure 11: les différents architectures RNN [22]

➢ *LSTM :*

C'est le type le plus répandu des réseaux de neurones récurrents et résout les problèmes de réseaux de neurones récurrents rencontrés précédemment,

Les LSTM(long short term memory) sont utiles pour se souvenir des informations pendant longtemps car ils stockent leur mémoire interne de la même manière qu'un ordinateur, ils lisent, écrivent et suppriment des informations selon les besoins grâce à l'utilisation de portes. Ces portes aident le réseau à décider quelles informations conserver et quelles informations supprimer de la mémoire (ouvrir ou non la porte) en fonction de l'importance attribué à chaque bit d'information. C'est ce qui est très avantageux, il permet non seulement de stocker plus d'informations (sous forme de mémoire à long terme), mais aide aussi à éliminer les informations inutiles qui peuvent modifier le résultat d'une prédiction, telles que les articles d'une phrase [23], la figure suivante montre un exemple de l'architecture LSTM :

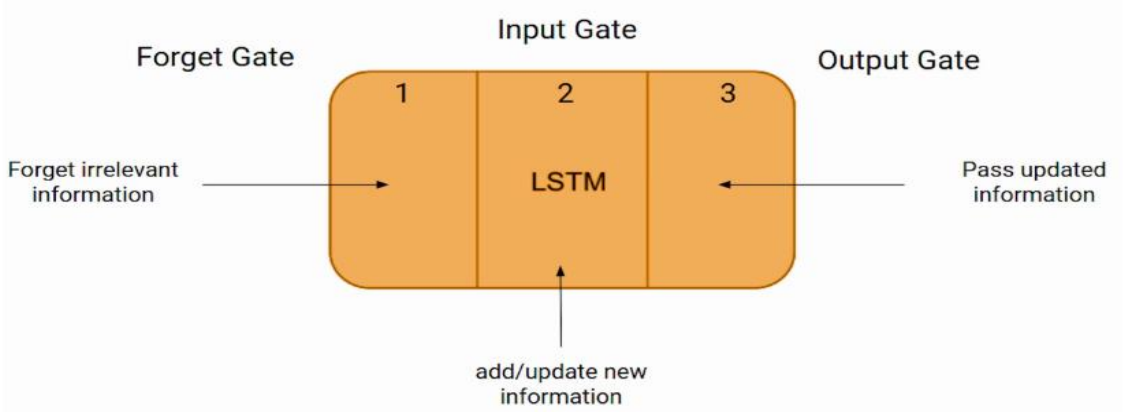


Figure 12 : architecture LSTM [24]

4.3. Transformer

Ce type d'architecture a été introduit en décembre 2017 par vaswani .al dans Google Brain et Google Research, s'appuyant sur les mécanismes d'attention et aucun réseau récurent ou convolutionnel [25] .

L'architecture du transformer est un empilement de 6 couches en entrée à gauche qui est l'encodeur et 6 couches en sorties à droite qui est le décodeur mais prenant chacun la sortie du 6eme encodeur.

Chaque entrée de l'encodeur de l'encodeur est la sortie du précèdent, l'entrée du premier encodeur est un vecteur embedding, le dernier décodeur est connecté à un bloc « Réseau de neurones linéaire + Softmax ».

L'encodeur est constitué de deux blocs de réseaux de neurones qui sont le « Self-attention » et un réseau à propagation avant, de son côté le décodeur est également composé d'un bloc de Self-attention et d'un Feed-forward et une couche « Encoder-Decoder Attention » qui a pour but de permettre au décodeur de réaliser le mécanisme d'attention entre la séquence d'entrée (encodée) et la séquence de sortie (en train d'être décodée), l'architecture suivante montre le modèle Transformer :

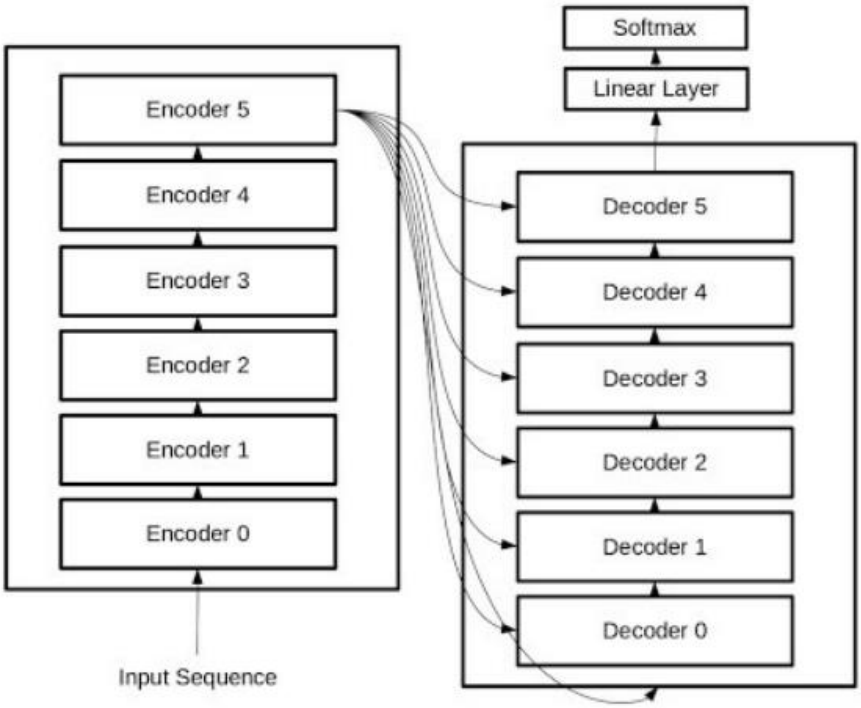


Figure 13: architecture Transformer [26]

Les mécanismes d'attention sont devenus une partie intégrante de la modélisation de séquences et des modèles de traduction dans diverses tâches, ils permettent la modélisation des dépendances sans tenir compte de leur distance dans les séquences d'entrée ou de sortie [27].

Le self-attention ou intra-attention est un mécanisme d'attention mettant en relation différentes positions d'une seule séquence afin de calculer une représentation de la séquence. L'auto-attention a été utilisé avec succès dans une variété de tâches, y compris la compréhension de la lecture, le résumé abstrait, implication textuelle …etc , un exemple est montré dans la figure suivante :

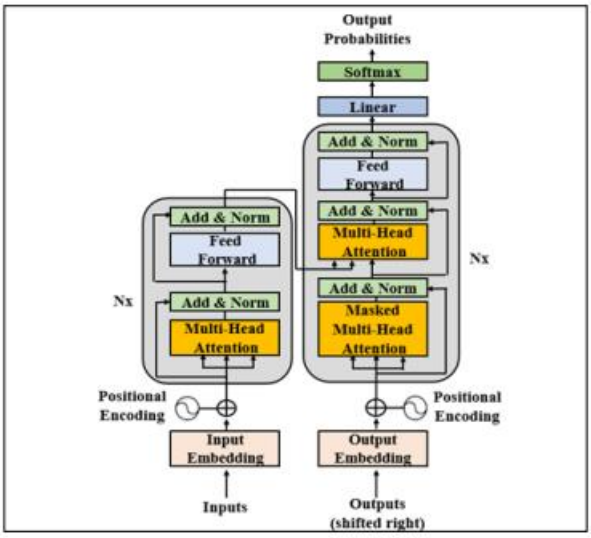


Figure 14: architecture Transformer [27]

4.4. Les autos encodeurs

Les auto encodeurs sont des réseaux de neurones qui ont pour objectif d'avoir une sortie plus proche de l'entrée, possède une architecture appelé bottleneck constitué d'un encodeur qui transforme l'entrée en une représentation dans un espace de dimension plus faible appelé espace latent ainsi l'entrée est compressée en une représentation moins coûteuse.

Le décodeur, a pour but de reconstruire à l'aide de la représentation latente de l'entrée, une sortie la plus fidèle à l'entrée.

L'encodeur apprend les composantes les plus importantes d'une entrée pour avoir la meilleure compression possible**,** afin que le décodeur ait les informations les plus essentielles pour reconstruire [28] .

L'apprentissage est donc « auto-supervisé » car la loss à minimiser est le coût de reconstruction entre la sortie et l'entrée , illustré dans la figure suivante :

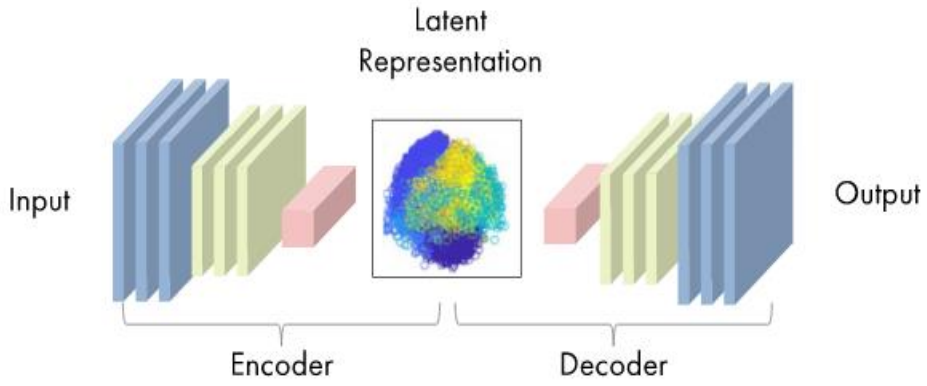


Figure 15: les auto encodeurs [29]

4.5. Les architectures génératives

4.5.1. Les réseaux antagonistes génératifs

Il représente une architecture pour l'apprentissage non supervisé très répandu dans le domaine de la vision par ordinateur, composé de deux modèles de réseaux de neurones le générateur et le discriminateur, ces deux réseaux s'entrainent simultanément sur un jeux de données [30] .

Au cours de l'apprentissage, le générateur basé sur un réseaux convolutif, prend un bruit en entrée réussit à créer des images, en parallèle le discriminateur essaie de distinguer les images fausses produites par le générateur, au fur et à mesure le générateur produit des images de plus en plus réel alors que le discriminateur finit par distinguer les images irréelles, la figure suivante montre l'architecture général du modèle GAN(generative adversial network) :

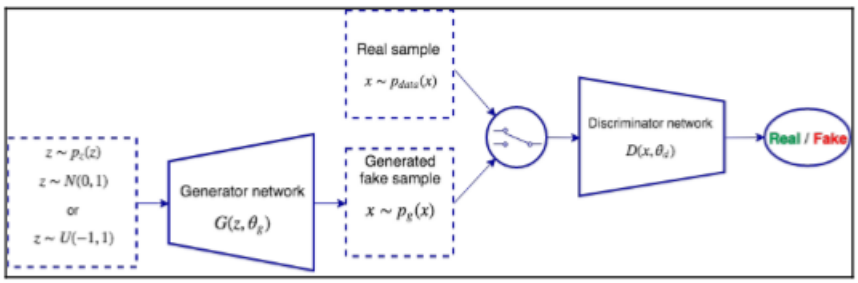


Figure 16: architecture GAN [31]

Mathématiquement, le discriminateur et le générateur jouent un rôle jeu minimax à deux joueurs avec la fonction valeur V (G, D) dont la fonction objective est :

$$V(G,D) = Ex \sim pdata[log(D(x))] + Ez \sim pz[log(1 - D(G(z)))])$$

D(x) : est la probabilité que les données x proviennent de l'ensemble d'apprentissage.

D(G(Z)) : est la probabilité que les données G(Z) générées sont fausses par le générateur.

Le générateur essaie pour minimiser V (en minimisant D(x) et D(Z)), le discriminateur essaie de maximiser V (en maximisant D(x) et en minimisant D(G(Z)).

4.5.2. Conditonal–GAN

Dans ce type de modèle le générateur et le discriminateur sont conditionnées par des informations supplémentaires appelés y qui pourrait être une ou n'importe quel caractéristique, pour effectuer un conditionnement, on alimente y à la fois dans le discriminateur et le générateur comme couche d'entrée supplémentaire [32] , comme illustré dans la figure suivante :

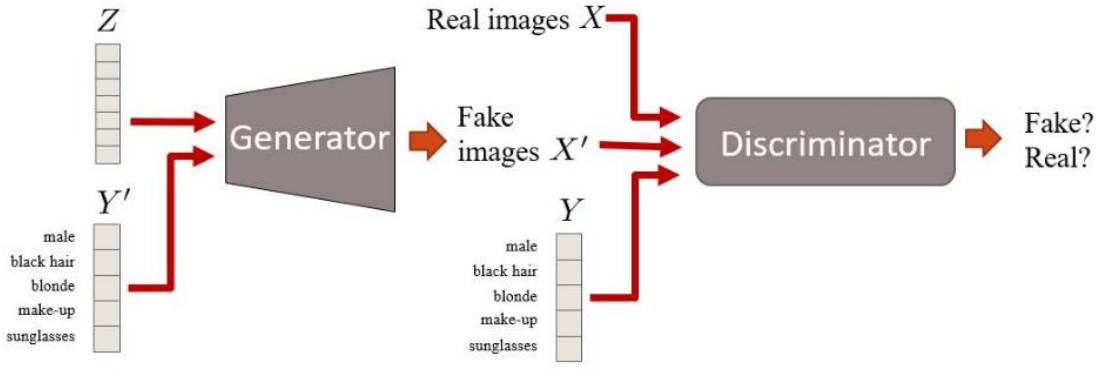


Figure 17: L'architecture CGAN [33]

4.5.3. Wasserstein GAN

Une extension des réseaux antagonistes qui a été introduit par Martin Arjovsky, et coll. dans leur article de 2017 dans le but de résoudre le problème de lenteur

d’entrainement des GANS classique, qui a l’instar de ce derniers le discriminateur classe les images généré par le générateur, l’idée du WGAN est de remplacer le discriminateur par une critique qui teste la réalité de l’image produite ,ainsi, au lieu de classer simplement les images d'entrée comme vraies ou fausses, le discriminateur W-GAN (ou critique) génère un score pour informer le générateur de la réalité ou de la fausseté de l'entrée image [34].

En utilisant KL divergence, la tache de WGAN sera de minimiser la divergence entre p_z et p_{data} à l’aide des différents mesures de distance tels que :la distance Wesseirstein ou Earth Mover (EM), la divergence de Kullback Liebler(KL),Jensen-Shannon(JS), la figure suivante montre un exemple de l’architecture WGAN :

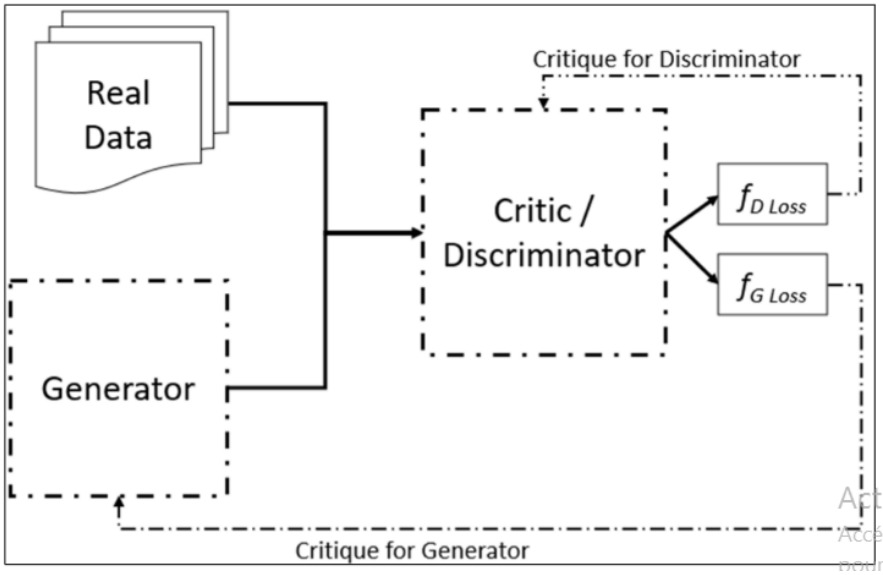


Figure 18: l'architecture WGAN [35]

5. Model et performance du Machine learning

5.1. La validation croisée

L’approche de base que l’on peut adopter est de créer un échantillon d’entraînement, sur lequel on va constituer le modèle et un échantillon de test, sur lequel on va tester le modèle.

Pour évaluer la qualité du modèle et sa performance en prévision, on utilise une métrique de performance P, l'idée est de prendre la qualité de l'ajustement ou de la prévision qui est calculée pour chaque jeux de données, à partir de la métrique [36].

En pratique, on prend souvent 60 % des données pour l'entraînement, 20 % pour la validation et 20 % pour le test.

5.2. Métrique de performance

5.2.1. Cas de régression

Nombreuses sont les mesures disponibles pour évaluer la qualité d'un modèle de régression. Elles se basent toutes sur des calculs réalisés à partir de trois grandeurs [36] :

• La valeur observée d'une série à prédire y_i

• La valeur prédite par le modèle pour cette même valeur observée $\widehat{y_i}$

• une prévision naïve de référence, qui est la moyenne de la valeur observée y^2.

Elles permettent de calculer, pour tout i des m observations

Ce qui permet de définir des indicateurs de performance du modèle, les plus connus sont les suivantes :

• l'erreur moyenne absolue (MAE, Mean Absolue Error) :

$$MAE = \frac{1}{m}\sum_{i=1}^{m}\left|y_i - \widehat{y_i}\right|$$

• la racine carrée de la moyenne du carré des erreurs (RMSE, Root Mean Squared Error) :

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \widehat{y_i})}$$

MAE et RMSE correspondent à une indication agrégée de l'erreur de prévision [36].

5.2.2. Cas de classification

L'évaluation d'un problème de classification se base sur une matrice de confusion, qui met en regard des données prédites et des données observées, comme le montre le tableau suivant :

		Observations		
		+	-	Total
Prédictions	+	250	150	400
	-	50	550	600
	Total	300	700	1000

Tableau 2: exemple de matrice de confusion [36]

(Les termes « + » /« - » peuvent être remplacés par toutes les sorties possibles d'un problème de classification binaire : vrai/faux, présent/absent, oui/non, sain/malade, etc.).

Cette matrice permet de calculer une première mesure très intuitive : le taux d'erreur, c'est-à-dire le taux de mauvaise classification (50 + 150) /1000 = 20 % (prédictions « - » alors que « + » plus prédictions « + » alors que « - », divisé par nombre total d'individus). Mais d'autres mesures peuvent être tirées à partir de cette matrice, afin de décrire plus généralement le comportement du modèle par rapport aux données réelles, pour cela, définissons les termes suivants :

		Observations		
		+	-	Total
Prédictions	+	Vrais positifs (VP)	Faux positifs (FP)	Total des positifs prédits (VP + FP)
	-	Faux négatifs (FN)	Vrais négatifs (VN)	Total des négatifs prédits (FN + VN)
	Total	Total des vrais positifs observés (VP + FN)	Total des vrais négatifs observés (FP + VN)	Taille totale de l'échantillon (N)

Tableau 3: Les termes définis par la matrice de confusion [36]

Ainsi, le taux d'erreur évoqué précédemment peut être défini par : (FN + FP) /N.

Tout un ensemble d'autres indicateurs peut être calculé à partir de ces mesures.

En général, pour évaluer un modèle, on utilise conjointement les deux indicateurs suivants :

• le taux de vrais positifs VP/ (VP + FN), aussi appelé rappel (recall) ou sensibilité

• la précision VP/ (VP + FP).

Pour comparer plusieurs modèles, on utilise également un indicateur agrégé, composé à partir du rappel et de la précision : le F1 score. On calcule pour cela la moyenne harmonique de la précision et du rappel (cela permet de pondérer les deux mesures de façon équivalente).

$$F_1 = \frac{2(précision * rappel)}{précision + rappel}$$

Autrement écrit, à partir des termes définis plus haut :

$$F_1 = \frac{2VP}{2VP + FP + FN}$$

6. Conclusion

Dans ce chapitre, nous nous sommes concentrés sur deux parties, l'apprentissage automatique et le deeplearning, nous avons expliqué les différentes architectures notamment le fonctionnement des CNN pour la classification d'images, les réseaux de neurones récurrent destiné à traiter les textes, les modèles de génération dont les réseaux antagonistes génératives et leurs variantes, nous avons aussi présenter les transformer.

Le prochain chapitre sera consacré aux travaux connexes de text to shape.

Chapitre 2 : travaux connexes

1. Introduction

Depuis les dernières années, la génération d'images prend un impact important dans le deep learning, des recherches performantes ont été faite dans le but de produire des images.

Toutefois, peu de travaux se focalisent spécialement sur la génération de formes 3D.

Dans ce chapitre nous survoleront les approches de génération d'images en général depuis des descriptions textuelles, puis nous présenterons les différentes manières de produire des formes avec leurs modalités.

2. Text to image

La synthèse Texte à l'image consiste à convertir des descriptions textuelles en images appropriées.

Actuellement, les modèles GAN sont largement utilisés pour de meilleurs résultats.

Nous nous sommes concentrés sur deux approches utilisées avec GAN.

Au cours des dernières années, de nombreux études se sont focalisé sur la génération des images à partir des descriptions textuelles Reed et .al sont les pionniers de l'approche text to image en utilisant la puissance des réseaux antagonistes génératives dans ce qui suit nous détaillerons deux travaux de la tache text to image.

2.1. Conditional-GANs

Nous prenons [37] comme exemple afin d'expliquer la technique, dans leur article, le principe de C-GAN est définis en mettant un vecteur one-hot class étiquetée comme entrée du générateur et du discriminateur en plus du vecteur de bruit échantillonné aléatoirement.

Il en résulte une stabilité d'entraînement plus élevée, des résultats plus attrayants visuellement, ainsi qu'un générateur contrôlable les sorties, le but de Générateur est de tromper le discriminateur alors que ce dernier consiste à identifier les données correctes, générateur et discriminateur sont à la fois rivaux.

Le générateur essaie de convaincre le discriminateur que les fausses instances générées sont les vrais échantillons de données et augmente également la probabilité d'erreurs, alors que les chiffres du discriminateur identifie les vrais des fausses.

Cependant, ces étapes sont répétées plusieurs fois et les sous-modèles s'entraînent de façon meilleure.

D'abord, le Discriminateur est formé sur les vrais échantillons de données pour vérifier s'il peut identifier ces échantillons comme réels [38].

Après il est entrainé sur de fausses données générées pour voir s'il est capable de discriminer entre image réelle et fausse.

Le générateur est également entrainé selon les résultats du discriminateur afin qu'il puisse s'améliorer.

En plus de construire de bons incorporations de texte(text embedding), la traduction du texte en images est hautement multimodale [37],le terme « multimodal » est important pour se familiariser avec les recherches DeepLearning, cela fait référence au fait qu'il existe de nombreuses images différentes des oiseaux et qui correspondent à la description textuelle « oiseau » [37].

L'apprentissage multimodal est également présent dans le sous-titrage d'image, (image-texte). Cependant, cela est grandement facilité en raison de la structure séquentielle du texte telle que le modèle peut prédire le mot suivant conditionné à l'image aussi bien qu'aux mots préalablement prédits.

L'apprentissage multimodal est traditionnellement très difficile, mais il est beaucoup plus facile avec l'avancement des GAN (Generative Adversarial Networks), ce framework crée une fonction de perte adaptative bien appropriée aux tâches multimodales comme le texte à l'image.

La figure ci-dessous montre l'architecture Reed et al. Utilisé pour former ce modèle GAN texte-image. Les plats à emporter les plus remarquables de ce diagramme est la visualisation de la façon dont l'incorporation de texte s'intègre dans la séquence traitement du modèle. Dans le réseau générateur, le texte incorporant est filtré à

travers une couche entièrement connectée et concaténé avec le vecteur de bruit z , dans ce cas, l'incorporation de texte est convertie à partir d'un format 1024x1 vecteur à 128x1 et concaténé avec le vecteur de bruit aléatoire z 100x1 sur le côté du réseau discriminateur, le text-embedding est également compressé à travers une couche entièrement connectée en un vecteur 128x1, puis remodelé en une matrice 4x4 et concaténée en profondeur avec la représentation de l'image.

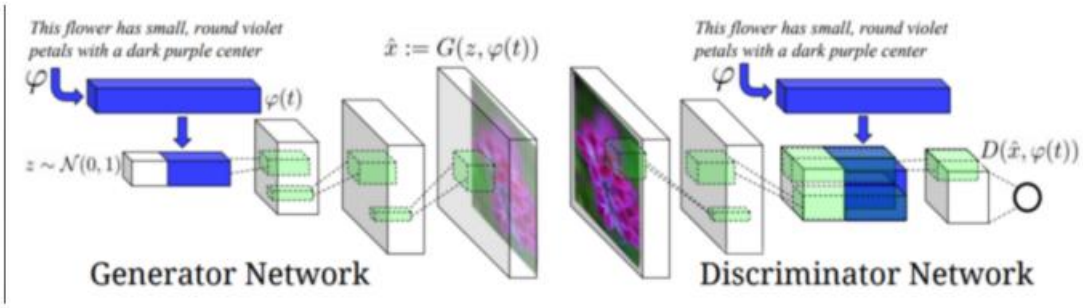


Figure 19: le modèle Reed et al. text to image [39]

Cette représentation de l'image est dérivée après que l'image d'entrée a été convoluée à plusieurs fois, la résolution spatiale est réduite et des informations seront extraites,cette stratégie d'intégration pour le discriminateur est différente du modèle conditionnel-GAN dont l'intégration est concaténée à la matrice d'image d'origine, puis convolué [40] .

Il est à noter que le diagramme d'architecture visualise comment le DCGAN sur échantillonne les vecteurs ou les images basse résolution pour produire des images haute résolution, on peut voir que chaque couche déconvolutive augmente la résolution spatiale de l'image [41].

De plus, la profondeur des caractéristiques des cartes diminuent par couche, on peut voir également comment les couche de convolution du réseau discriminateur diminue la résolution spatiale et augmente la profondeur des caractéristiques des cartes lors du traitement de l'image.

Lors du processus d'entrainement il est difficile de séparer la perte de l'image générée qui n'a pas l'air réaliste ou la perte de l'image générée qui ne correspond pas à la description du texte.

Les auteurs de l'article décrivent l'entrainement dynamique dont initialement le discriminateur ne prête aucune attention au texte embeddings, puisque les images créées par le générateur ne semblent pas du tout réelles [38] . Une fois que G peut générer des images qui passent au moins le critère réel vs faux, alors le text embedding est aussi pris en compte.

Les auteurs suppriment l'entraînement dynamique en rajoutant des paires d'images réelles avec une description de texte incorrect étiquetées comme "fausses".

Le discriminateur se concentre uniquement sur la tâche binaire du vrai contre le faux et ne considère pas séparément l'image en dehors du texte. Ceci est en revanche une approche telle que AC-GAN avec étiquettes de classe encodées à chaud (one-hot encoded class labels).

Le discriminateur AC-GAN produit du vrai par rapport au faux et utilise un classificateur auxiliaire partageant les caractéristiques intermédiaires pour classer les étiquette de classe de l'image [42].

2.2. DF-GAN

Depuis quelques temps, d'autres modèles ont vu le jour qui adoptent une architecture empilée et qui emploie cross modal attention pour fusionner texte et caractéristiques de l'image et introduisent DAMSM network [43] , cycle consistance [44] ou SIAMSE network [45] pour assurer la consistance sémantique text-image.

Cependant ,cross-modal attention ne peut pas tirer pleinement parti des informations textuelles, des problèmes ont été survenus , les architectures empilés présentent des enchevêtrement entre les différents générateurs , ce qui a produit une image assez floue avec quelque détails , en outre les études menés par TingTing Qiao et al .corrigent les réseaux supplémentaires lors de l'entrainement, d'autre part des études existantes corrigent généralement les réseaux supplémentaires lors de l'entrainement

adverse , ce qui rend ces réseaux facilement dupés par le générateur pour synthétiser les caractéristiques adverses [46] .

L’architecture DeepFusion-GAN proposée par les auteurs Ming Tao et al. qui utilise la puissance des modèles génératives pour la génération d’image réalistes à partir d’une description textuelle , backbone one-stage qui est un seul générateur remplacera stacked–backbone [47] , composé d'une hinge-loss [48] et de réseaux résiduels, dans le but de stabiliser le processus d'entraînement GAN pour synthétiser directement des images haute résolution, puisqu’il n'y a qu'un seul générateur dans one-stage backbone ,ce qui évite les enchevêtrements entre différents générateurs.

Pour le deuxième problème, les auteurs proposent Target-Aware Discriminateur composé de Matching-Aware Gradient Penalty (MA-GP) qui représente une stratégie de régularisation sur le discriminateur et One-Way Output qui prédit le loss de l’adversaire directement, pour améliorer la cohérence sémantique texte-image.

MA-GP [49]construit une surface de perte lisse aux points de données réels et correspondants, ce qui encourage davantage le générateur à synthétiser des images de correspondance de texte.

Pour le troisième problème, un bloc Deeptext-imageFusion Block (DFBlock) est proposé pour fusionner plus efficacement les informations textuelles dans les caractéristiques de l'image.

Le DFBlock empile plusieurs transformations Affine et Couches ReLU dans Fusion Block. Pour la transformation Affine, les auteurs adoptent deux MLP (Multilayer Per ceptron) pour prédire le canal conditionné par la langue paramètres de mise à l'échelle

DFBlock représente un module léger et qui manipule les caractéristiques visuelles des carte grâce à une opération de mise à l'échelle et de décalage par canal.

L'empilement de multiples DFBlocks à toutes les échelles d'image approfondit le processus de fusion texte-image et réalise une fusion complète entre le texte et les caractéristiques visuelles.

Le DF-GAN proposé est composé d'un générateur, d'un discriminateur et d'un encodeur de texte préformé, comme illustré à la figure suivante :

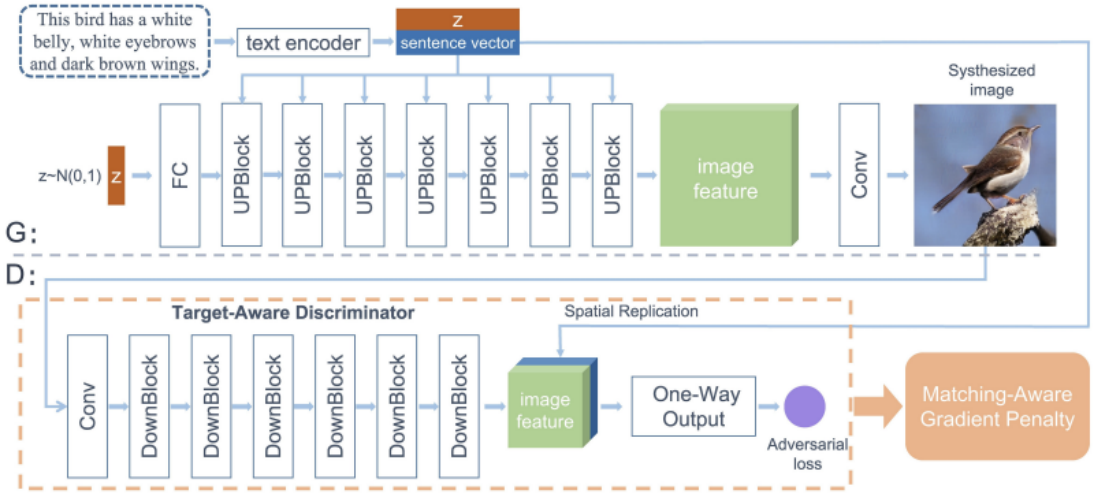


Figure 20: l'architecture du modèle DFGAN [46]

Le générateur a deux entrées, un vecteur de phrase codé par un encodeur de texte et un vecteur de bruit échantillonné à partir de la distribution gaussienne pour assurer la diversité des images générées.

Le vecteur de bruit est d'abord introduit dans une couche entièrement connectée et remodelé, ensuite une série de UPBlock lui sera appliqué pour suréchantillonner les caractéristiques de l'image.

L'UPBlock est composé d'une couche de sur échantillonnage, d'un bloc résiduel et de DFBlocks pour fusionner les caractéristiques du texte et de l'image pendant le processus de génération d'image.

Enfin, une couche de convolution convertit les caractéristiques de l'image en images.

Le discriminateur convertit les images en caractéristiques d'image via une série de DownBlocks qui est composé d'une couche de sous-échantillonnage et un bloc résiduel.

Ensuite, le vecteur phrase sera reproduit et concaténé avec les caractéristiques de l'image.

Une perte contradictoire sera prédite pour évaluer le réalisme visuel et la cohérence sémantique des entrées.

En distinguant les images générées des échantillons réels, le discriminateur encourage le générateur à synthétiser des images avec une meilleure qualité et une cohérence sémantique texte-image. L'encodeur de texte est une mémoire bidirectionnelle à long court terme (LSTM) [50]qui extrait les vecteurs sémantiques de la description du texte.

Pour évaluer l'architecture DF-GAN deux types de datasets ont été entrainés CUBbird [1]et COCO [2],

Pour entrainer le modèle les auteurs utilisent l'optimizer Adam, avec un taux d'apprentissage de 0.0001 pour le générateur et 0.0004 pour le discriminateur.

L'évaluation de la performance du réseau proposé, les auteurs choisissent IS (inception score) et FID (Frechet Inception Distance).

FID est l'une des mesures les plus utilisées pour mesurer la distance entre les points caractéristiques des images réelles et des images générées [51] .

IS calcule le Kullback-Leibler (KL) divergence entre une distribution conditionnelle et une distribution marginale, si l'IS est plus élevé cela signifie une meilleure qualité des images générées.

Les auteurs de l'architecture DF_GAN ont comparé les deux datasets COCO et CUBbird en utilisant les indices de performance FID et ID par rapport aux différents architectures GAN en se focalisant sur ceux du texte-image, il s'avère que l'IS ne peut bien évaluer la qualité des images du dataset COCO et à la place de l'IS les auteurs optent pour le nombre de paramètre (NoP).

Les auteurs de DF-GAN comparent avec l'architecture Attn-GAN qui emploie une attention intermodale pour fusionner les caractéristiques du texte et de l'image, ils

[1] L'ensemble de données CUB contient 11 788 images appartenant à 200 espèces de oiseaux, chaque image d'oiseau a dix descriptions linguistiques

[2] L'ensemble de donnée COCO contient 80 000 images pour la formation et 40k images à tester, chaque image de cet ensemble de données comporte cinq descriptions de langue.

ont trouvé que DF-GAN améliore la métrique IS de 4,36 à 5,10 et diminue la métrique FID de 23,98 à 14,81 sur l'ensemble de données CUB et par rapport aux architectures Mirror-GAN [52] et SD-GAN [53] qui utilisent la cohérence du cycle et le réseau siamois pour assurer la cohérence sémantique texte-image, l'IS de DF-GAN augmente de 4.56 et 4.67 à 5.10, idem sur le jeu de données CUB.

Cependant, DF-GAN Comparé à l'architecture DM-GAN [54] introduit Memory Network pour affiner le contenu flou de l'image, l'IS s'améliore de 4,75 à 5,10 et le FID diminue de 16,09 à 14,81 sur le dataset CUB, et FID de 32,64 à 19,32 sur le dataset COCO.

D'âpres les résultats précédents, les auteurs ont atteint leurs objectifs avec succès.

3. Image en forme

Dans l'article [55],les auteurs ont proposé une nouvelle approche pour la génération des formes 3D basé sur un réseau génératif antagoniste qui génère les visages humains et qui consiste en deux phases :

La première consiste à générer des images stylé de dessin animés à partir des images de visage généré par GANs ce qui permet de manipuler les expressions faciales [55] .

La deuxième est de reconstruire les formes 3D à partir des images 2D.

La génération des images des visages humains a été utilisé via Style-GAN.

Les auteurs proposent les méthodes d'inversion GAN dans le cas où l'entrée est une image de visage du monde réel afin de projeter l'image de visage donnée dans l'espace latent W via le générateur de visage, dans le but d'obtenir le code latent correspondant, une traduction d'image à image sera effectuée pour ajouter un style de dessin animé aux images de visage d'origine. À cette fin, le modèle FFHQ pré-entraîné sera affiné pour former un modèle de génération de visage de dessin animé, les mêmes codes latents seront transmis à la fois au dataset FFHQ et au modèle de génération de dessin animé, ils affirment que cela permet à ces deux modèles de génération de partager un espace latent W similaire, ce qui fait que les images de dessin animé et de visage humain générées à partir des mêmes codes latents ont des attributs sémantiques similaires.

Le framework proposé par les auteurs peut être résumé comme un schéma d'apprentissage en trois étapes :

• Étape 1. Ils ajustent FFHQ pré-entraînée en modèle Style-GAN avec l'ensemble de données de dessins animés. Ensuite, les poids du modèle de génération de dessins animés seront interpolés pour générer des visages de dessins animés photo réalistes 2D.

• Étape 2. Les auteurs cherchent à découvrir les directions sémantiques de l'espace latent Style-GAN pour manipuler un seul concept sémantique des images générées, tout en gardant l'identité du visage inchangée.

• Étape 3. Les auteurs utilisent Style-GAN pour donner des pseudo-échantillons pour reconstruire les formes 3D des visages de dessins animés, qui sont générés sur la base des codes latents originaux et découverts.

Les travaux récents de Karras et al. [56] et ceux de Wang et al. [57] ont démontré que Style-GAN peut produire des images de haute-fidélité et l'espace latent appris, est désenchevêtré en fonction de divers attributs sémantiques, permettre de manipuler les codes latents et générer des images de visage avec différentes expressions faciales, poses et conditions d'éclairage utiles pour la reconstruction de formes 3D.

Pour générer le model de cartoon 2D , les auteurs utilisent la technique d’apprentissage par transfert et la technique d'interpolation de modèle pour former un modèle Style-GAN sur l'ensemble de données de dessin animé, en prenant le modèle Style-GAN [58]pré-entraîné sur FFHQ comme modèle de base qui sera ensuite affiné sur le dessin animé, ils adoptent la technique d'interpolation qui permet au modèle Style-GAN de produire des images de dessins animés plus photo réalistes.

La reconstruction des formes 3D selon les auteurs, le module (V, L, D, A) est nécessaire pour prédire le point de vue, l'éclairage, la profondeur et l'albédo.

Pour une image donnée, les auteurs mentionnent qu’il faut prédire les modules cités en haut.

Dans la première étape, le réglage canonique sera utilisé pour les conditions d'éclairage et de point de vue.

Ensuite les auteurs échantillonnent au hasard les différents points de vue et conditions d'éclairage qui sont utilisés ensemble avec la profondeur et l'albédo calculés à la première étape pour générer les images rendues. A la fin les auteurs mentionnent qu'il faut former conjointement les réseaux (V, L, D, A), ces étapes seront répétés quatre foi pour affiner les résultats reconstruits en 3D.

4. Texte en forme 3D

Dans l'article [59] , les auteurs représentent une méthode pour générer les formes 3D à partir du langage naturel pour cela ils divisent leur travail en deux tâches majeures récupération du texte à mettre en forme et génération de texte à mettre en forme.

Tout d'abord ils présentent une méthode pour l'apprentissage en conjointant texte et espace de représentation de forme directement à partir des descriptions en langage naturel des instances de forme 3D ,En exploitant un nouvel ensemble de données de descriptions appariées en langage naturel et des formes 3D colorées, leur méthode s'étend l'apprentissage par association [60] et apprentissage métrique [61] pour apprendre conjointement le texte et le 3D forme embedding qui regroupent formes et descriptions similaires, établissant des connexions sémantiques implicites suivi de leur cadre de génération de texte en forme, contrairement aux travaux connexes dans synthèse texte-image [62] [63],ils ne reposent pas sur des étiquettes de classe ou pré-entrainement sur de grands ensembles de données.

En outre, ils entrainent le texte et les composants formes codés (shape encodings) conjointement de bout en bout, associant des points similaires dans les données à la fois au sein d'une modalité (texte à texte ou forme à forme) et entre les deux modalités (texte à forme).

La tâche de récupération permet d'évaluer la qualité text-shape embedding conjointement appris par rapport aux travaux précédents.

Quant à la tâche de génération de texte en forme, Kevin et al. S'est concentré sur la génération de formes colorées car la plupart des descriptions de formes impliquent la couleur ou les propriétés des matériaux.

Pour s'acquitter de cette tâche, ils ont combiné les modèles embedding conjointement avec un nouveau framework conditionnel et Wasserstein GAN, offrant une meilleure qualité et diversité de sortie par rapport à un conditionnel -GAN formulé.

Enfin, ils ont utilisé vector embedding arithmétique et le générateur pour manipuler les attributs des forme [64].

Résultats expérimentaux sur 75 000 descriptions en langage naturel collectées pour 15 000 formes de chaises et de tables dans l'ensemble de données ShapeNet [65] montrent qu'elles modélisent et surpassent par une grande marge en ce qui concerne les tâches de récupération et de génération, la figure suivante explique l'approche proposé de Kevin et al .

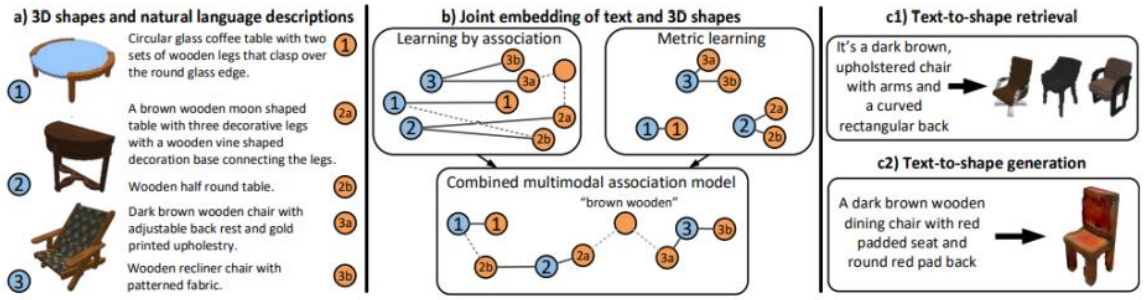


Figure 21: l'approche text to 3D shape de Kevin et al. [59]

5. Conclusion

Dans ce chapitre, nous avons expliqué trois principaux travaux connexes : Text to image, Image to shape, et texte to 3D shape.

Pour la génération des images à partir du texte nous avons choisi l'architecture Conditional-GAN dont nous avons détaillé le principe, nous avons aussi parlé de l'architecture DFGAN récemment utilisé, pour la transformation des images en formes, nous avons pris en compte le travail dernièrement publié sur la génération des cartoon 3D à partir des images en utilisant STYLE-GAN et enfin le text to shape qui est la base de notre travail tel que ce dernier utilise l'architecture de CWGAN.

Dans le chapitre suivant, nous passerons à la partie conception, qui sera consacrée à expliquer les architectures et les méthodes utilisées dans la réalisation de notre approche.

Chapitre 3 : conception et réalisation

1. Introduction

Ce chapitre représente la partie conceptuelle où nous expliquerons la méthode proposée pour la génération des formes 3D.

Dans un premier lieu nous explorerons le Word-embedding dédié et nous allons exposer la méthode BERT, ensuite nous parlerons de la génération des formes 3D et nous expliquerons le méthode WGAN intégré à notre approche pour la génération des formes.

2. L'architecture globale

Notre approche consiste en deux blocs : l'encodage et la génération, l'encodeur prend en entrée la description textuelle et génère son embedding en utilisant la méthode BERT qui sera ensuite concaténé à un bruit et passeront au générateur qui produit la forme 3D associé à la description textuelle.

La forme 3D en sortie sous forme de grille de voxel sera ensuite encodé et convertie vers un vecteur avec shape encoder et sortira ensuite vers un réseau critique avec la description textuelle, la critique est connectée à deux fonctions de perte qui n'est autre que ***Mean Squared Error.***

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(y_n - \widehat{y_n})^2$$

Une pour la formation du générateur si la critique a bien classé la forme et l'autre pour la formation du discriminateur si la critique a mal classé la forme, le réseau critique calculera la distance wesserstein, le générateur tente de minimiser la distance en retro propagation de l'erreur.

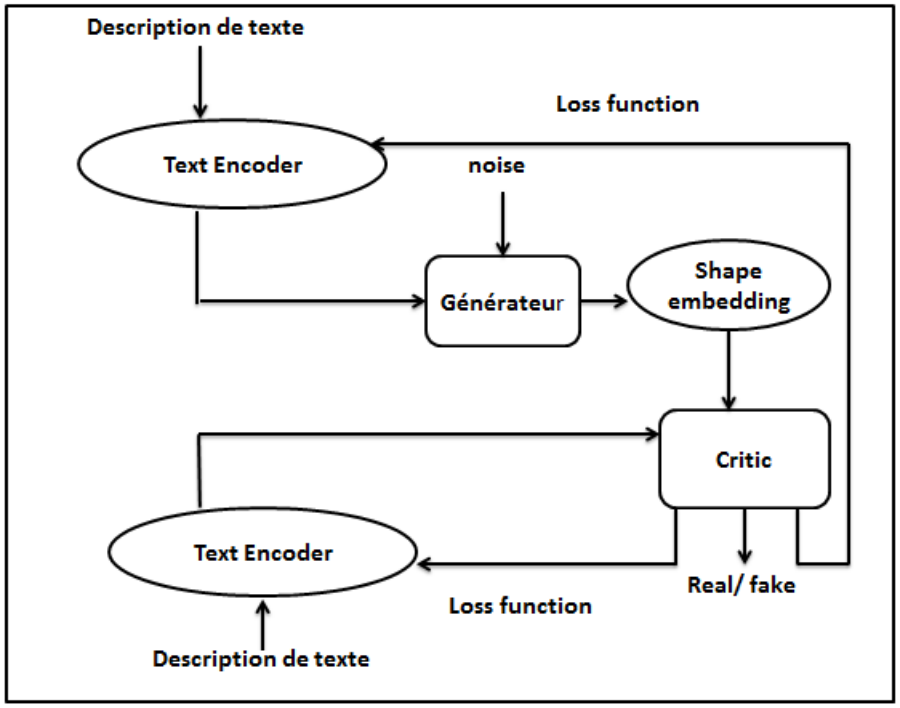


Figure 22: l'architecture générale de notre approche

3. Traitement du texte

L'objectif principale du traitement de la description textuelle dans les NLP est de réduire la dimension en la convertissant en un vecteur numérique pour cela il existe deux approches : le one-hot-encoding et le wordembedding.

3.1. One- hot encoding

Dans ce type d'approche chaque mot est représenté dans un vecteur de longueur n et n qui représente la longueur de tout le vocabulaire [66] .

Le vocabulaire est le nombre total de mots uniques dans le document, prenons une phrase simple avec son vecteur codé comme le montre la figure suivante :

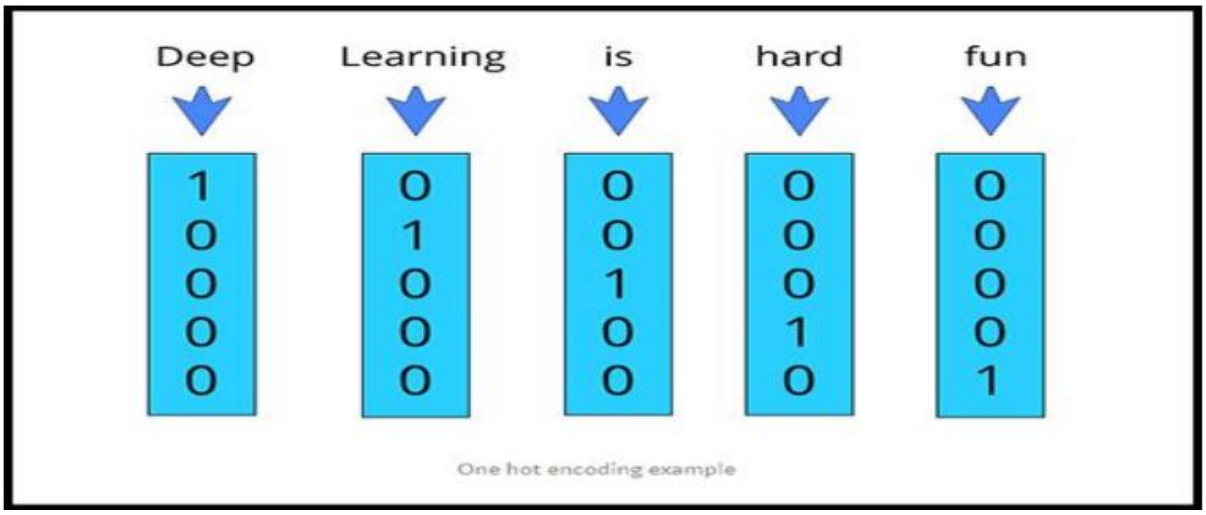


Figure 23: exemple d'encodage à chaud [67]

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

3.2. Word embedding

C'est un type qui est très populaire pour représenter des données textuelles dans des problèmes résolus par des algorithmes d'apprentissage en profondeur.

Il fournit une représentation dense d'un mot rempli de nombres flottants.

La dimension vectorielle varie selon le vocabulaire Taille. Il est courant d'utiliser un mot incorporé de dimensions 50, 100, 256, 300 et parfois 1 000. La taille de la dimension est un hyper-paramètre avec lequel nous devons jouer pendant la phase de formation. Si nous essayons de représenter un vocabulaire de taille 20 000 en représentation one-hot, nous nous retrouvons avec 20 000 x 20 000 nombres, dont la plupart seront zéro. Le même vocabulaire peut être représenté dans word embedding par 20 000 x taille de dimension, où la dimension la taille peut être 10, 50, 300, créer des word embedding consiste à commencer avec des vecteurs denses pour chaque jeton contenant des nombres aléatoires, puis entraînez le modèle [68].

Il existe de nombreux approches pour générer des incorporations de mots, nous nous concentrerons sur BERT car il est utilisé dans notre approche.

3.2.1. Bert embedding

Bert a été développé et introduit par l'équipe Google en 2018, il a été très performant pour la résolution des taches NLP, il est de type Transformer mais se limite à un encodeur [69] .

Il existe deux types de modèles préformés BERT :

- Base BERT : 12 couches, 768 tailles cachées, 12 têtes d'auto-attention, 110M paramètres.
- BERT Large : 24 couches, 1024 tailles cachées, 16 têtes d'auto-attention, 340M paramètres.

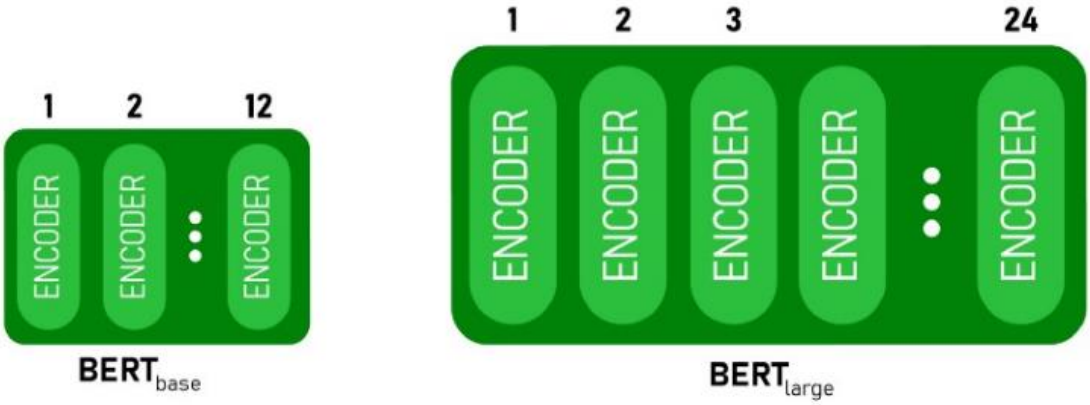


Figure 24: représentation graphique des modèles BERT [70]

Le processus de BERT utilise deux techniques de pré entrainement :

- *Masked Language Model (MLM)*ou chaque mot d'une phrase a une probabilité de 15 % d'être masqué et le modèle est entraîné à prédire les mots masqués. Par exemple, si la phrase d'origine est « Elle s'est amusée à la fête d'anniversaire», alors le modèle peut recevoir «Elle <masque>amusée à la <masque>d'anniversaire » et doit prédire les deux mots masqués <s'est> et <fêtes>en ignorant les autres mots , chaque mot sélectionné à 80 % de chances d'être masqué, 10 % de chances d'être remplacé par un mot aléatoire et 10% de chances d'être écarté.

- *Next Sentence Prediction (NSP)* : l'entrée est composée de deux phrases, deux nouveaux tokens sont rajoutés :
    - [CLS] est un jeton de classification binaire ajouté au début de la première séquence pour prédire si la deuxième séquence suit la première séquence.
    - [SEP] est un jeton de séparation qui signale la fin d'une séquence.

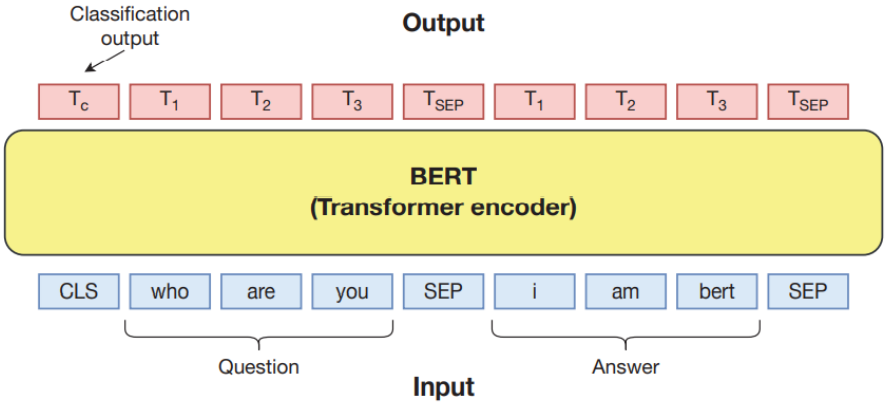


Figure 25: modèle BERT entrée et sortie

La couche d'entrée est simplement le vecteur des jetons de séquence avec les jetons spéciaux, BERT utilise WordPiece pour la tokenisation qui divise le jeton comme "gathering" en "gather" et "ing". Les intégrations de jetons sont les identifiants de vocabulaire pour chacun des jetons et une intégration de phrase est juste une classe numérique pour distinguer les phrases A et B.

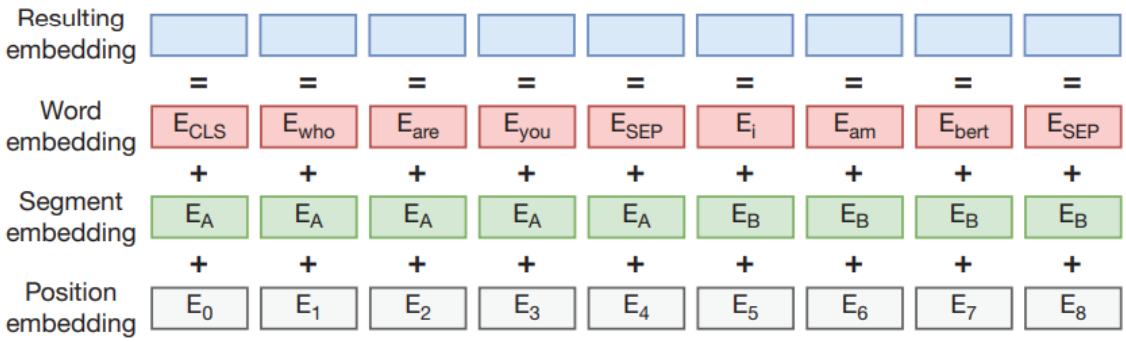


Figure 26: la représentation des sorties avec BERT [71]

4. Génération de notre Text-Embedding

En ce moment ,all-MinLM-L6-V2 est un modèle transformer de phrase qui fait correspondre les phrases à un vecteur d'espace de 384 dimensions qui pourrait être utilisé dans les taches de regroupement et la recherche sémantique , il est destiné à être utilisé comme un encodeur de phrase et paragraphe court, tout en créant un vecteur de sortie de l'input texte qui capture l'information sémantique, par défaut les phrases contenant plus de 256 mots seront coupés, l'une des raisons pour laquelle nous avons choisi ce modèle c'est qu'il s'adapte parfaitement aux titre de notre jeu de données , la figure suivante montre ses principales caractéristiques :

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 27: all-MiniLM-L6-v2 model information

On commence par télécharger le modèle, puis nous appelons la fonction (encode ()) sur le captions du dataset et génère l'intégration de chaque captions, le processus peut prendre un certain temps mais reste parmi les plus performants, il est le plus rapide (sur un GPU V100, il encode 14200 phrases par seconde) [72].

5. Traitement des formes 3D

Pour gérer les formes 3D dans les taches de machine Learning qui sont assez couteuse , nous pouvons générer l'embedding capturé par les principaux caractéristiques de nos formes en se concentrant sur les propriétés les plus importants qui distinguent chaque forme d'une autre et en ignorant tout ce qui est répétitif et non schématique, on propose une architecture autoencodeur qui possède deux blocs encodeur et décodeur , l'encodeur encode la forme dans les embedding souhaités

alors que le décodeur obtient ces incorporations et décode pour avoir la forme original .

Leur architecture est simple, le codeur est composé de deux couche conv3d (3 filtre tridimensionnel) suivie à la fois d'une fonction d'activation RELU (qui produit la valeur si elle est positive, sinon elle produira 0) le premier utilise un noyau de (32, 32, 32) et un stride de (2,1,1) avec padding= (4, 2, 0) tandis que le second convoluté à travers la sortie de la première couche avec un noyau de taille (5, 5, 1) sans stride et sans padding (tous deux définis sur 1). Tandis que le décodeur applique un opérateur de convolution transposé 3D sur les embedding générés par le codeur, qui se font par deux couche transposées à convolution avec les mêmes paramètres dans l'ordre inverse (la première couche utilise un noyau de (5,5,1) et pas de stride ou de padding et le premier noyau= (32, 32, 32), stride =(2, 1, 1), padding =(4, 2, 0)).

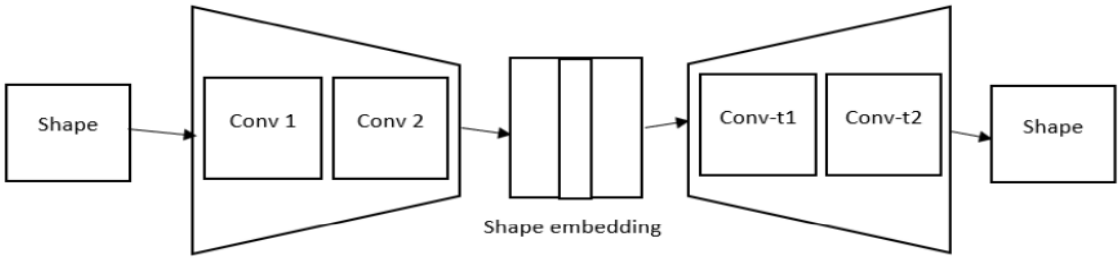


Figure 28: notre architecture auto encodeur

On applique mean squared error loss pour améliorer le modèle autoencodeur entre la forme généré par le décodeur et la forme en entrée dans l'encodeur et nous mettrons à jour les poids tout au long du processus de retro propagation, nous entrainons le modèle pour 100 époques à l'aide d'un lot de 4 formes à la fois.

6. La tâche de génération

Nous avons utilisé trois variantes des GANs pour cette tâche (GANs, WGAN, CGAN), dans les réseaux antagonistes génératifs deux blocs sont utilisés : le générateur et le discriminateur pour comprendre et générer des données complexes à l'aide d'opérations mathématiques.

Il existe deux cas possibles pour chaque architecture pour entrainer le modèle, nous aurons besoin de la description qui sera associé avec la forme correspondante.

Dans le premier cas, les formes 3D seront encodés avec un autoencoder et dans le second cas, on utilisera directement la forme.

Pour nos modèles à base de GAN, on y introduira une description textuelle représentée sous forme vectorielle obtenue depuis le modèle BERT. Afin d'entrainer un GAN, nous auront également besoin d'un vecteur de bruit pour générer des formes 3D imitée (non réelles).

Nos modèles GAN seront composés de deux parties : un générateur et un discriminateur.

6.1. Le générateur

Dans le cas du WGAN , le générateur est un modèle séquentiel de 5 couches linéaires qui sont, mathématiquement, conçues pour calculer l'équation linéaire Ax = b où x est l'entrée, b est la sortie, A est le poids et on applique la transformation linéaire à l'entrée donnée pour avoir une autre taille , ce qui nous permet de passer d'un vecteur latent de taille 384 à une forme de (4,32,32,32) à la fin, les quatre premières couches sont suivies d'une couche de batch normalization de 1 dimension, à l'exception de la première, et chacun des quatre utilise une fonction d'activation d'unité linéaire rectifiée(Leaky Relu) . La dernière couche produit le shape embedding de taille (4,3,3,3) et utilise la fonction d'activation de Tanh. Bien que nous ayons l'intention de générer des formes, nous avons pensé qu'il serait préférable de générer directement les intégrations de formes, nous les avons essayées toutes les deux, mais la génération avec l'embeddings était bien meilleure en termes de temps et de performances, donc la génération avec le shape embedding obtient le vecteur latent de taille 384 et ressorte l'embedding de taille (4,3,3,3).

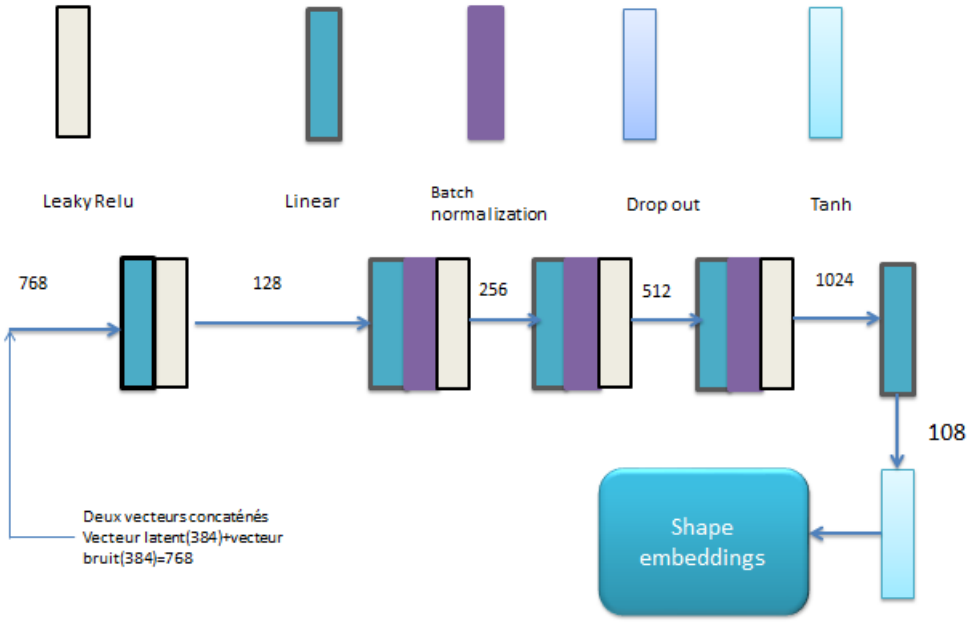


Figure 29: l'architecture du générateur de WGAN

Par contre l'architecture GANs avec les mêmes paramètres n'est pas très performante car elle génère des formes aléatoires, puisqu'il n'existe pas de vecteur latent.

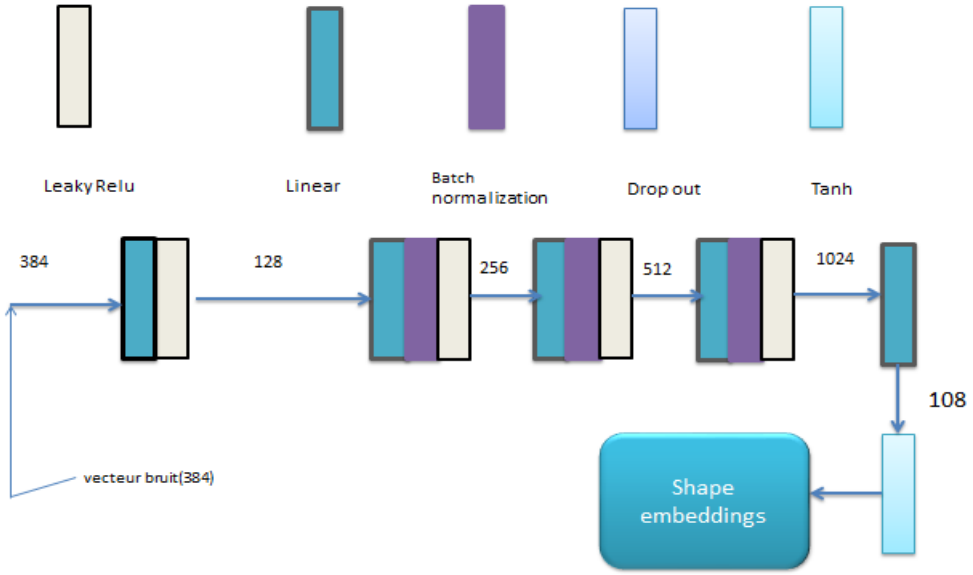


Figure 30: l'architecture du générateur GANs

Concernant l'architecture CGAN avec les mêmes paramètres donne les mêmes résultats du générateur WGAN.

6.2. Le discriminateur

De l'autre côté , dans le discriminateur contient 4 couches linéaires, la première on a comme caractéristique d'entrée 492 et sort une carte de caractéristiques de 512 celle-ci sera transmise à travers des unités linéaires rectifiées (Leaky Relu) et à travers deux couches linéaires avec une fonction d'abandon de probabilité de 0,4 ce qui fait qu'elle garde le même taille du vecteur mais met à zéro certaines de ses valeurs avec la probabilité de 40 % d'un élément est mis à zéro, à la fin, le vecteur principalement mis à zéro est alimenté à une couche linéaire qui produit un score qui est une métrique de régression(Mean squared error) ,s'il s'agit d'une fausse ou d'une vraie forme, et améliorée dans le processus de rétro propagation après chaque époque et de cette façon, il motive le générateur à intensifier son jeu et générer des formes difficiles pour que le discriminateur fasse la distinction entre eux et les vrais, c'est ainsi que nous obtenons le générateur final qui peut générer des formes proches du réel à partir des descriptions textuelles.

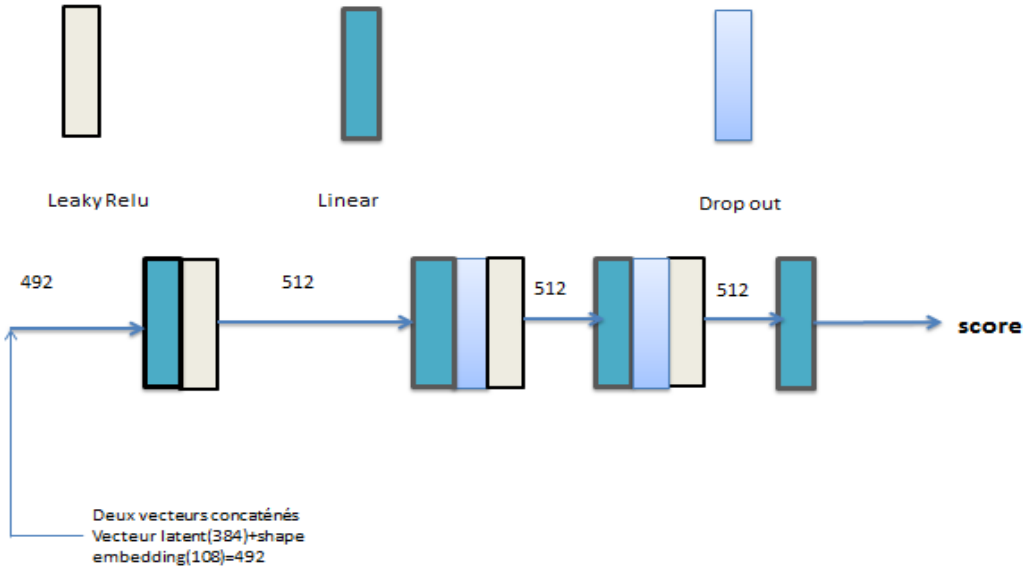


Figure 31: l'architecture du la critique de WGAN

Les mêmes paramètres ont été utilisé dans les discriminateurs des trois architectures sauf que dans l'architecture GANs le discriminateur arrive à classifier les formes 3D imitée de celle qui sont réel et donne les mêmes performances

Dans le cas de l'architecture CGANS le discriminateur à la couche final produit une valeur déterminant la forme si elle est fausse ou vrai, et la précision de la

classification sera améliorée dans le processus de rétropropagation après chaque époque.

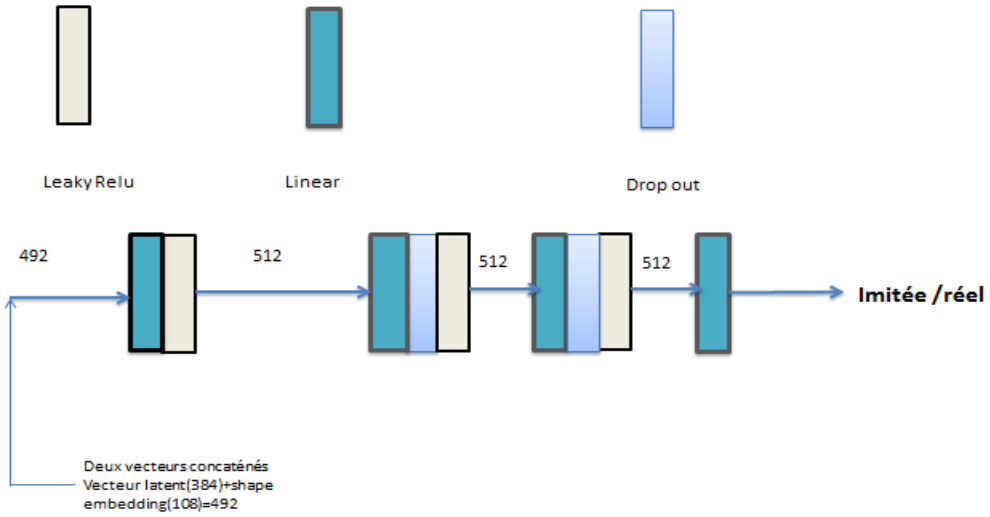


Figure 32: architecture du discriminateur de CGAN

7. Conclusion

Dans ce chapitre nous nous somme penché sur l'approche proposé, nous avons expliqué en détails la méthode BERT pour le wordembedding, nous avons aussi montré la méthode d'encodage des formes 3D et enfin nous avons exposé notre architecture en comparant avec les autres variantes GANs.

Dans le prochain chapitre, nous parlerons de outils utilisés avec l'évaluation de notre travail.

Chapitre 4 : Test et résultat

1. Introduction

Dans le chapitre précèdent nous avons expliqué l'architecture du modèle proposé, dans ce qui suit nous allons présenter les outils utilisés et nous finirons par une brève évaluation.

2. Les outils utilisés

Nous citons dans ce qui suit les différents outils utilisés pour notre réalisation

2.1. Google collabotary

Connu sous le nom de Colab est un environnement de bloc-notes Jupyter gratuit qui s'exécute dans le cloud et stocke ses blocs-notes sur Google Drive., il permet d'écrire et d'exécuter Python dans un navigateur, sans avoir recours à une configuration au préalable , accès gratuit aux GPU et partage facile, depuis qu'il utilise google drive on peut facilement lire les données du lecteur dans Colab et l'utiliser, ce qui est l'un des meilleurs fonctionnalités disponibles pour les développeurs du monde entier.

2.2. Python

Python est un langage de programmation qui est devenu un incontournable de la science des données, permettant aux analystes de données et à d'autres professionnels d'utiliser le langage pour mener calculs statistiques complexes, créer des visualisations de données, créer des algorithmes d'apprentissage automatique, manipuler et analyser des données et effectuer d'autres opérations liées aux données Tâches. Python peut créer une large gamme de visualisations de données différentes, comme la ligne et des graphiques à barres, des camemberts, des histogrammes et des tracés 3D. Python a aussi un nombre de bibliothèques qui permettent aux codeurs d'écrire des programmes pour l'analyse de données et la machine apprendre plus rapidement et efficacement, comme TensorFlow, Pytorch et Keras.

2.3. Outils de développement front-end

Pour notre interface nous avons utilisé HTML, CSS et JS, dans l'interface il y a un texte zone donc l'utilisateur peut mettre une petite description décrire la forme qu'il

veut, et cliquez sur générer la description traitée dans notre backend et générer la forme.

2.4. F3D

F3D est un programme de bureau créé par c++ et peut afficher des formes 3D dans NRRD format (le format principal que nous utilisons dans notre travail). Après que notre modèle a généré les formes nous glissons et déposons le fichier NRRD généré dans le programme et il l'affiche.

2.5. Visual Studio Code

C'est un éditeur de code simplifié prenant en charge le développement opérations telles que le débogage, l'exécution de tâches et le contrôle de version. Il vise à fournir juste les outils dont un développeur a besoin pour un cycle rapide de code-construction-débogage et laisse workflows plus complexes vers des IDE plus complets

2.6. NumPy

NumPy est un projet open source visant à permettre le calcul numérique avec Python. Il a été créé en 2005, en s'appuyant sur les premiers travaux du Numeric and Bibliothèques Numarray.

2.7. NRRD

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

2.8. Bert

Les représentations d'encodeurs bidirectionnels à partir de transformateurs (BERT) sont un technique d'apprentissage automatique basée sur les transformateurs pour la pré-formation en traitement du langage naturel (TAL) développée par Google. Il est conçu pour pré-entraîner en profondeur les représentations bidirectionnelles à partir

de texte non étiqueté en conditionnant conjointement contexte à gauche et à droite. En conséquence, le modèle BERT préformé peut être affiné avec une seule couche de sortie supplémentaire pour créer des modèles à la pointe de la technologie. Pour un large éventail de tâches.

2.9. Pytorche

PyTorch est une bibliothèque d'apprentissage automatique open source basée sur la bibliothèque Torch, utilisé pour des applications telles que la vision par ordinateur et le traitement du langage naturel, principalement développé par le laboratoire de recherche sur l'IA de Facebook (FAIR). Bien que le Python l'interface est plus raffinée et l'objectif principal du développement, PyTorch également possède une interface C++.

2.10. Datasets

ShapeNet est un vaste référentiel riche en informations de modèles 3D. Il contient modèles couvrant une multitude de catégories sémantiques, il fournit de vastes ensembles d'annotations pour chaque modèle et les liens entre les modèles du référentiel et d'autres données multimédia hors du référentiel.

ShapeNet fournit une vue des données contenues dans une catégorisation hiérarchique, contrairement à d'autres modèles référentiels, il fournit également un riche ensemble d'annotations pour chaque forme et les correspondances entre les formes. Les annotations comprennent les attributs géométriques tels que les vecteurs d'orientation verticale et avant, les pièces et les points clés, les symétries de forme et l'échelle de l'objet dans les unités du monde réel. Ces attributs fournissent des ressources précieuses pour traiter, comprendre et visualiser les formes 3D selon de la sémantique de la forme.

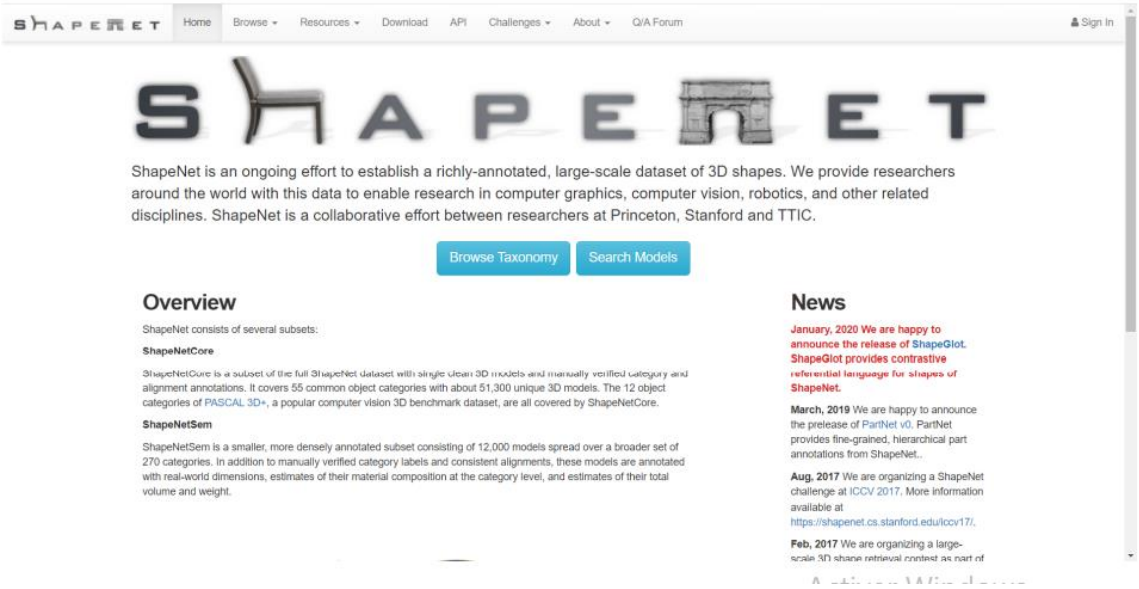


Figure 33: page d'accueil ShapeNet [72]

Pour créer un dataset réel avec de vrais objets 3D et un langage naturel descriptions, nous utilisons les catégories d'objets de table et de chaise ShapeNet (avec 8 447 et 6 591 cas, respectivement).

Ces formes 3D ont été créées par des concepteurs pour représenter avec précision des objets réels, un exemple du dataset shapenet est montré dans la figure suivante :

Figure 34: exemple de dataset ShapeNet table et chaise [72]

Nous avons choisi les catégories de table et de chaise car elles contiennent de nombreuses positions avec des variations d'attributs fines dans la géométrie, la couleur et le matériau. Nous augmentons cet ensemble de données de formes avec 75 344 descriptions en langage naturel (5 descriptions en moyenne par forme) fournies par des personnes sur Amazon et un dataset contrôlé et généré de manière procédurale de primitives géométriques 3D. Cet ensemble de données à grande échelle fournit de nombreuses descriptions complexes en langage naturel associées à des formes 3D réalistes, exemple illustré dans la figure suivante :

Figure 35: les formes avec leurs descriptions [72]

Pour permettre une évaluation quantitative systématique de notre modèle, nous utilisons un ensemble de données de primitives géométriques 3D avec les descriptions textuelles correspondantes. Ces données a été généré en voxélisant 6 types de primitives (cuboïdes, ellipsoïdes, cylindres, cônes, pyramides et tores) en 14 variations de couleurs et 9 variations de taille, les variations de couleur et de taille sont soumises à des perturbations aléatoires générant 10 échantillons de chacune des 756 configurations primitives possibles, créant ainsi 7560 formes voxélisées.

Ils ont ensuite créé des descriptions textuelles correspondantes avec une approche basée sur un modèle qui remplit les mots d'attribut pour la forme, la taille et coloriez

plusieurs ordres pour produire des phrases telles que « un grand cylindre rouge est étroit et haut ». Au total, nous générons 192 602 descriptions, pour une moyenne d'environ 255 descriptions par configuration primitive. Un tel texte synthétique ne ne correspond pas au langage naturel, mais il permet une référence facile avec un mappage clair aux attributs de chaque forme primitive comme illustré dans la figure suivante :

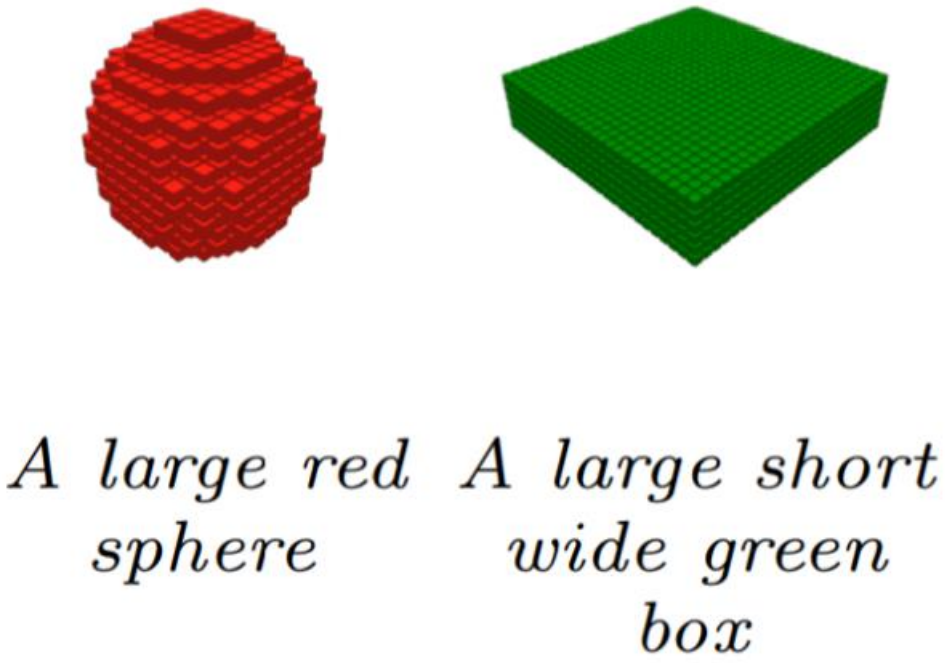


Figure 36: exemple de dataset primitive [72]

3. Interface principale

nous avons utilisé une interface Web pour que les utilisateurs interagissent et utilisent notre apprentissage en profondeur modèles, est une façon dont notre fin est une application simple qui a deux pages celui qui a le champ de saisie où un utilisateur peut entrer sa requête qui est dans notre cas une description textuelle, ce texte est intégré puis alimenté directement dans le générateur qui va générer la forme qui a été décrite textuellement par la requête de l'utilisateur et envoyer un tenseur un fichier nrrd, que nous faisons glisser et déposer dans le Interface F3D qui le visualisera sous forme de fichier 3D.

Ci-dessous des captures d'écran de la page principale de l’application.

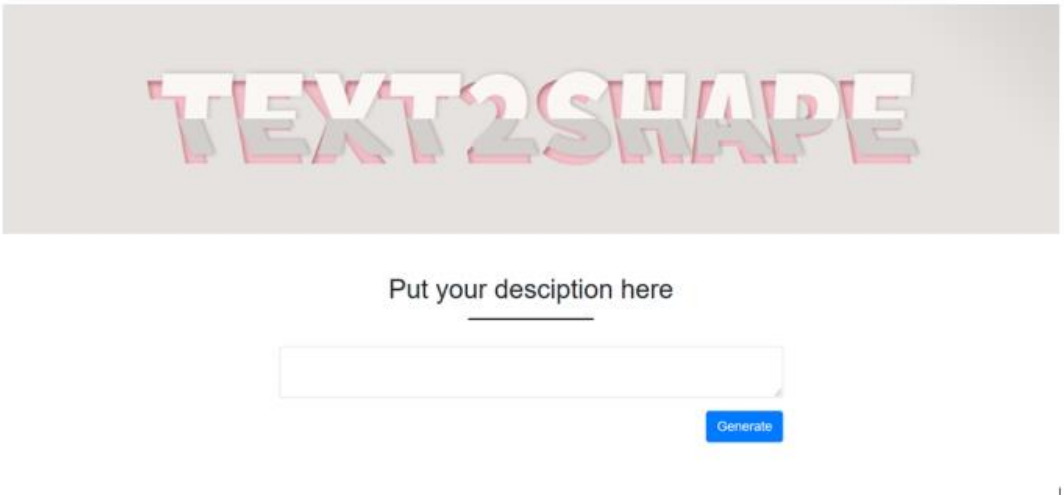


Figure 37: capture écran de la page d'accueil de l'application

4. Expériences

Dans cette partie, nous jetterons un coup d’œil sur les diffèrent architectures utilisés dans notre travail pour avoir les meilleurs résultats possibles.

4.1. Autoencoder

Nous avons utilisé un autoencoder pour la tache de shape embedding avec les paramètres suivants :

- Nombre d’epochs=100
- Batch size=8
- Le nombre de worker=4
- Optimizer=Adam
- Learning rate=e-3

La capture suivante montre l’entrainement de l’autoencoder sur 100 epochs.

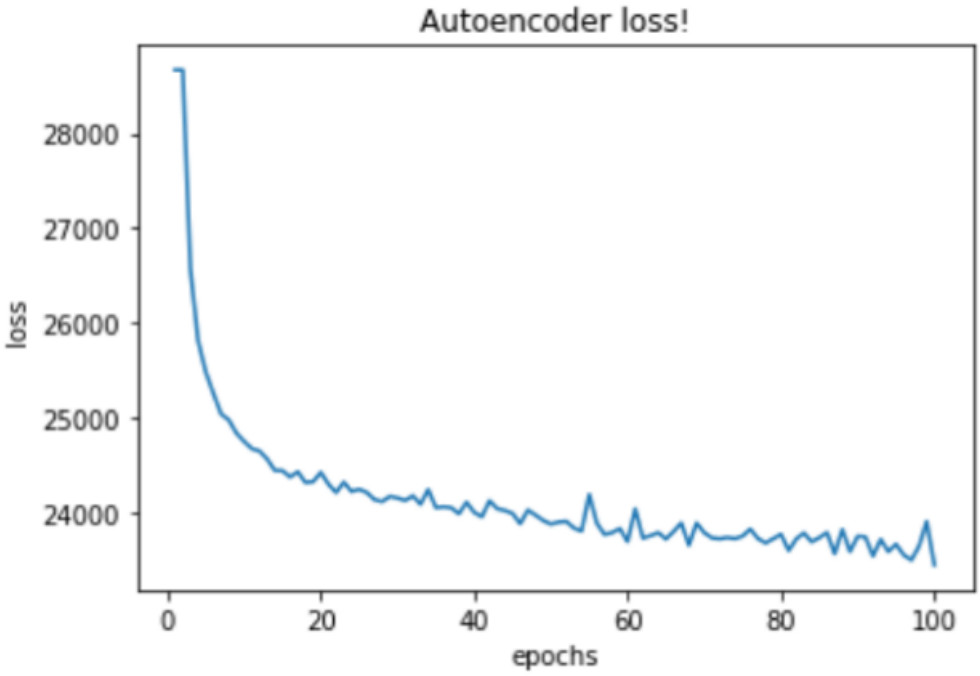


Figure 38: fonction de perte de l'autoencodeur

4.2. CGAN

Nous remarquons que dans les réseaux de neurones génératives conditionnel, le générateur génère les formes 3D sur la base des embeddings des descriptions textuelles concatènes avec le vecteur de bruit, les formes 3D seront introduit dans le discriminateur dans un premier cas avec le shape embeddings réalisé à travers l'encodeur et le deuxième cas sans l'utilisation d'un autoencodeur avec les paramètres suivants :

- Nombre d'epochs :200
- Batch size :32
- Nombre de worker=4
- Learning rate =0.0002
- Latent vecteur=384
- Shapes embedding = (4,3,3,3)

Dans ce qui suit les capture écrans des fonctions de perte avec et sans shape embedding :

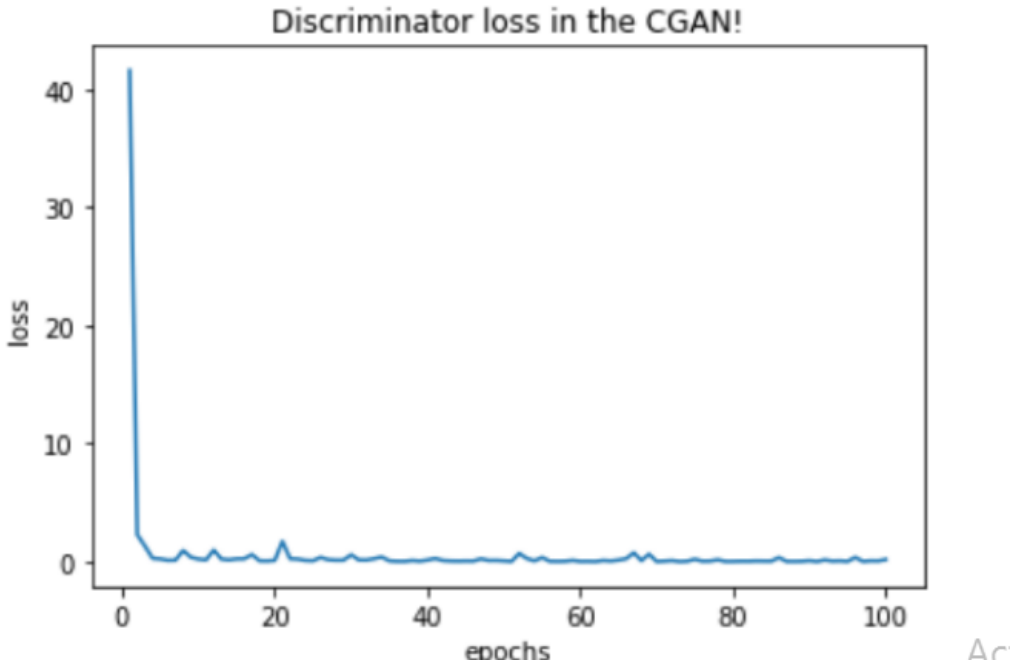


Figure 39: fonction de perte du discriminateur

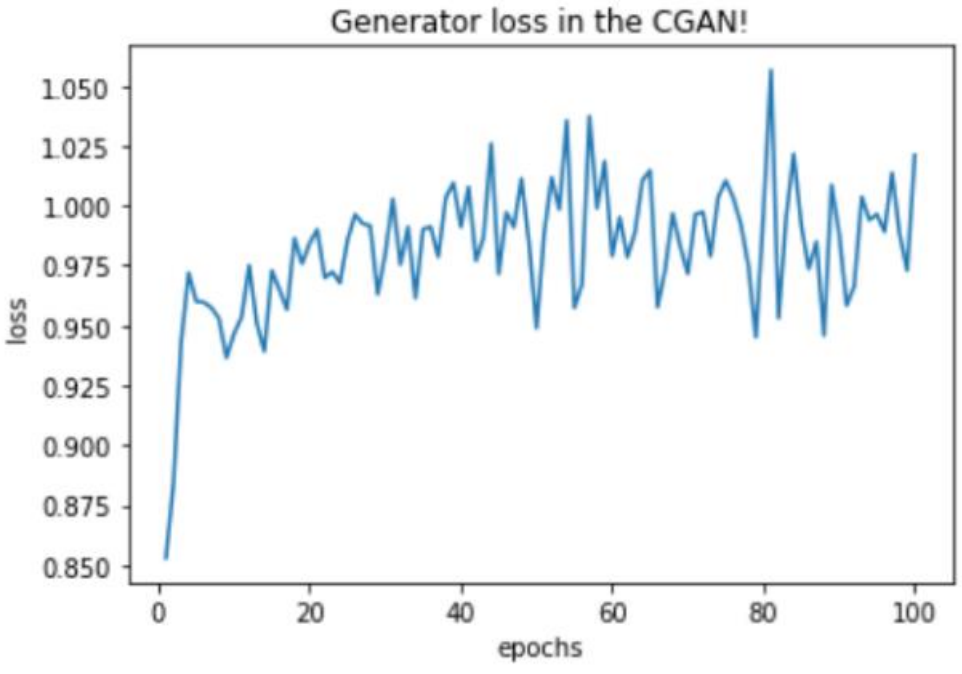


Figure 40: fonction de perte du générateur

Avec l’utilisation du shape embedding :

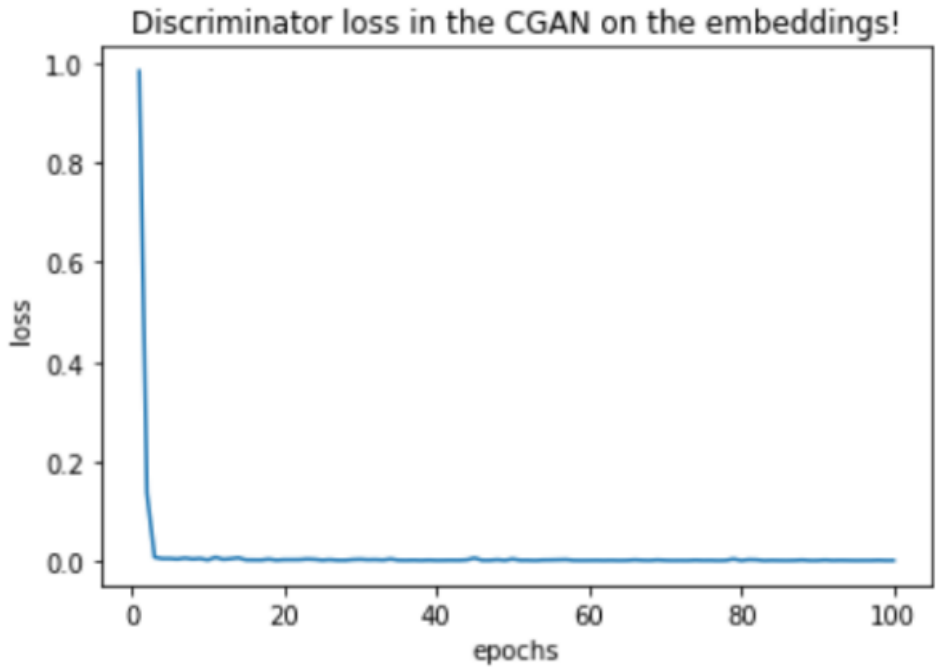


Figure 41: fonction de perte du discriminateur avec l'embedding

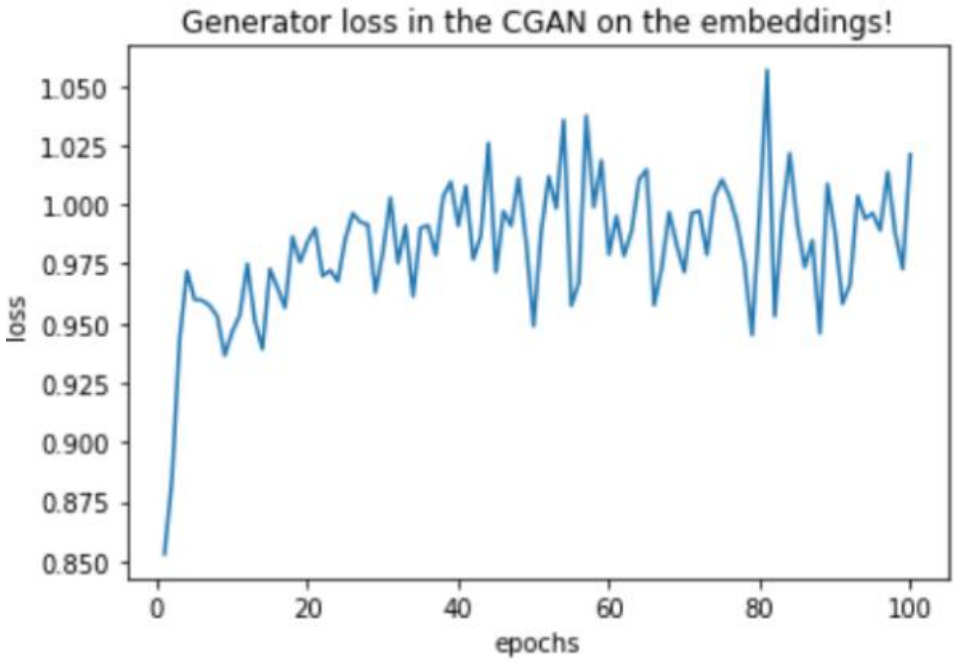


Figure 42: fonction de perte du générateur avec l'embeddings

5. Résultat

Selon les résultats obtenus on remarque que l'apprentissage du générateur avec les formes 3D sans les shape embedding donnent de meilleurs résultats pas une très grande marge mais qui peut être très utile.

De l'autre côté, le discriminateur avec le shape embedding surpasse d'une légère marge le discriminateur sans le shape embedding.

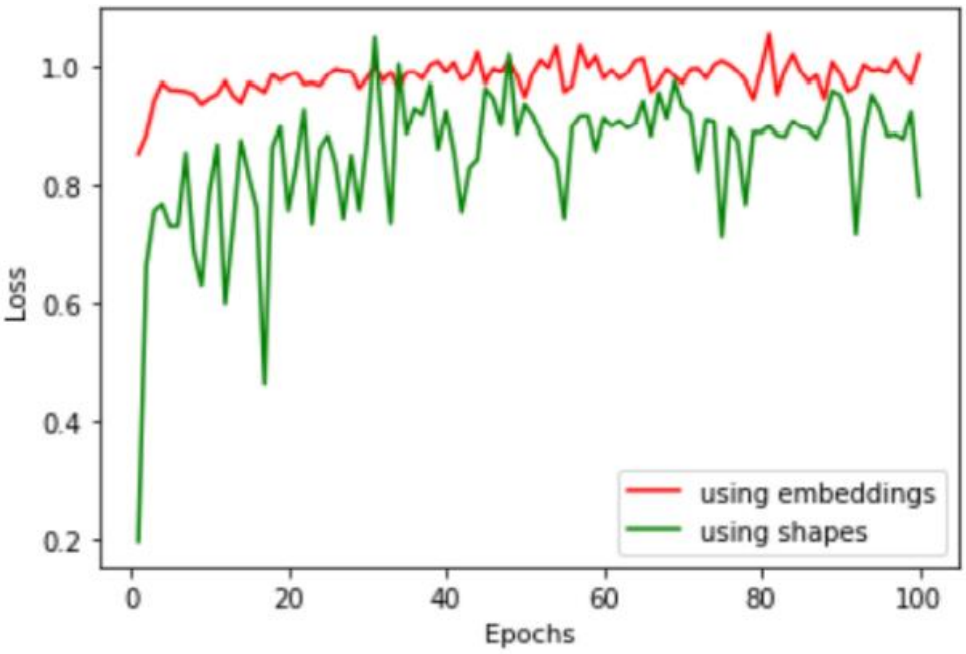


Figure 43: étude comparative du générateur

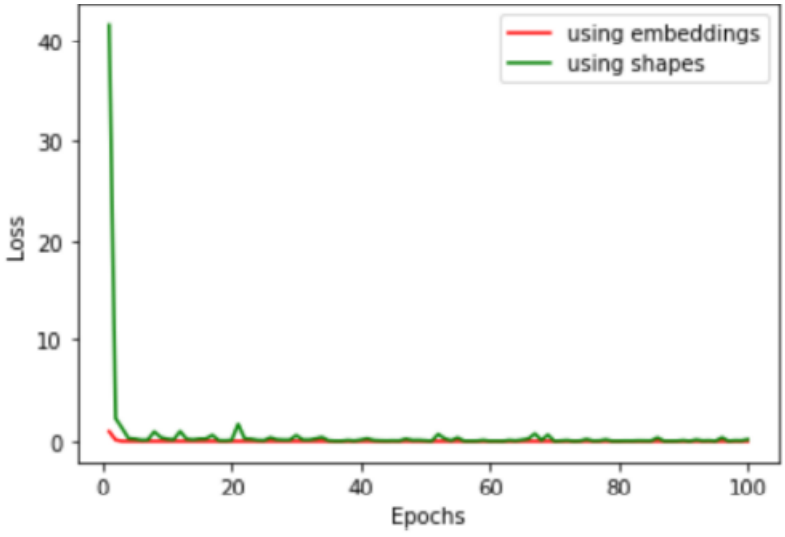


Figure 44: étude comparative du discriminateur

6. Discussion

Nous avons remarqué que le générateur donne de meilleur résultat avec les formes 3D originales contrairement au discriminateur qui est performant en utilisant le shape embedding pour la simple raison est la différence de taille entre la forme et son embedding , lors du traitement de la forme original , le générateur possède une multitude de façons de tromper le discriminateur contrairement à l'embedding qui possède une taille minimale résultat la critique peut avoir plus de chance de donner un score pour les formes fausses .

En pratique, nous avons besoin du générateur pour mieux fonctionner, même s'il faut trouver un équilibre entre le générateur et le discriminateur c'est le générateur qui va générer les formes pour nous. maintenant dans la pratique réelle, compte tenu du coût d'utilisation du formes qui, dans notre cas, ont pris plus de 4 heures par époque en comparaison avec l'embedding qui n'a fait le travail qu'en quelques minutes avec des exigences matérielles minimales pour faire les deux.

7. Conclusion

Ce chapitre a été consacré à la présentation des différents outils de développement de l'approche ainsi que le dataset utilisé dans l'application pour la génération des formes 3D, nous avons aussi discuté des différents résultats retrouvés.

Conclusion général

Conclusion générale

Pour finir ce mémoire, il est nécessaire de rappeler que la génération des formes 3D est une tâche très peu abordée par les chercheurs, ils se sont focalisés beaucoup plus sur les travaux text to image et image to shape 3D, mais rare sont ceux qui ont combiner les deux approches citées.

Nous avons pu réaliser notre travail en utilisant les formes 3D(chaise et table) du dataset Shapnet avec leurs description textuels , nous avons généré l'embedding des mots de la description en se servant de la méthode Google's BERT qui est l'une des taches du NLP ,le développement s'est fait dans l'environnement Google colab , nous avons implémenté un autoencodeur pour réduire le volume des formes 3D qui extrait les embeddings afin d'acquérir plus de performance , le but essentiel était d'utiliser une autre variante des GANs qui est le WGAN dont le générateur génère des formes 3D en se basant sur les descriptions textuelle mais qui contrairement aux CGANs utilise une critique et envoie un score pour déterminer les formes réel et irréels, nous avons ensuite discuté les résultats des trois architectures GANs CGAN et WGAN .

Grace à ce thème nous avons pu approfondir nos connaissances sur le deeplearning, nous avons pu découvrir leurs puissances dans les différentes taches de l'IA, nous avons pu aborder l'une des architectures les plus difficile pour la génération qui est les réseaux antagonistes génératifs

Parmi les perspectives de ce projet, il est possible de tester notre approche sur un autre type de dataset autre que Shapenet, utiliser d'autres formes 3D, améliorer le résultat des formes 3D en utilisant une autre méthode pour le word embeddings notamment GPT, il est aussi possible d'utiliser d'autres architectures des réseaux antagonistes génératifs.

Pour conclure, ce travail est dédié aux concepteurs de formes 3D et les designers pour faciliter leurs taches afin de gagner du temps et avoir plus de performance.

Références bibliographiques

Références Bibliographiques et webographie

[1] Guillaume Saint Cirgue, Apprendre le machine learning en une semaine., 2019.

[2] Judith Hurwitz and Daniel Kirsch, machine learning for dummies. River St. Hoboken: IBM, 2018.

[3] Aurélien Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems, Nicole Tache, Ed. USA: O'REILLY, 2019.

[4] Scott V. Burger, Machine learning avec R pour une modelisation rigoureuse. Paris , France: First, 2018.

[5] Shai Ben-David Shai Shalev-Shwartz, UNDERSTANDING MACHINE LEARNING From Theory to Algorithms, Cambridge University Press ed. New York, 2014.

[6] tariq rachid, make your own neural network. UK.

[7] https://lebigdata.fr/reseaux-de-neurones-artificiels-definition

[8] A.Traynard, D'une methode de gradient conjugué precondtionné element par element. le chensey France: institut national de recherche en informatique et en aeronotique , 1990.

[9] https://askabiologis.asu.edu/neuron-anatomy

[10] Warren S. McCulloch & Walter Pitts, "A logical calculus of the ideas immanent in nervous activities," University of Chicago, Chicago, 1943.

[11] http://aiandi.fr/comment-fonctionne-un-reseau-de-neurones/

[12] Jeff Heaton, Introduction to the Math of Neural Networks.: Heaton Research, 2012.

[13] Delip Rao & Brian McMahan, Natural Language Processing with PyTorch. United States of America: O'Reilly Media, 2019.

[14] Hyatt Saleh, The Deep Learning with PyTorch Workshop Build deep neural networks and artificial intelligence applications with PyTorch, Packt ed. Birmingham , Uk, 2020.

[15] Aurélien Géron, Deep Learning avec Keras et Tensorflow. Malakoff: DUNOD,

2019.

[16] Ragav Venkatesan and Baoxin Li, Convolutional Neural Networks in Visual Computing, Taylor & Francis Group, Ed. London , 2018.

[17] Hossein Rahmani,Syed Afaq Ali Shah,Mohammed Bennamoun Salman Khan, A Guide to Convolutional Neural Network for Computer Vision , Gérard Medioni Sven Dickinson, Ed. University of Toronto , Canada : the Morgan & Claypool Publishers, 2018.

[18] Aurelien V, Intelligence artificielle vulgarisé le machine learning et le deep learning par la pratique.: Eni, 2019.

[19] Ahmed Fawzy Gad, Practical Computer Vision Applications Using Deep Learning with CNNs With Detailed Examples in Python Using TensorFlow and Kivy. Egypt, 2018.

[20] https://medium.com/swlh/fully-connected-vs-convolutional-neural-networks-813ca7bc6ee5

[21] Eugene Charniak, Introduction auDeep Learning. Malakoff : Dunod, 2021.

[22] https://datascientest.com/recurrent-neural-network

[23] Rimsha Goomerb, Ashutosh Kumar Singh Jitendra Kumara, "Long Short Term Memory Recurrent Neural Network (LSTM-RNN)," in 6th International Conference on Smart Computing and Communications, kurkushetra india, 2017, p. 7.

[24] https://larevueia.fr/quest-ce-quun-reseau-lstm/

[25] Denis Rothman, Transformers for Natural Language Processing Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more, Safis Editing ed. Mumbai, 2021.

[26] https://towardsdatascience.com/a-deep-dive-into-the-transformer-architecture-the-development-of-transformer-models-acbdf7ca34e0

[27] Noam Shazeer,Niki Parmar Ashish Vaswani, "Attention Is All You Need," Google Brain , article 2017.

[28] *https ://blog.keras.io/building−autoencoders −in−keras.html*

[29] https://fr.mathworks.com/discovery/autoencoder.html

[30] Eli Stevens Luca Antiga Thomas Viehmann, deep learning with Pytorch, Manning Publications Co ed., 2020.

[31] Daniel Slater, Gianmario Spacagna, Peter Roelants ,Valentino Zocca Ivan Vasilev, Python Deep LearningExploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, PACKT ed. Birmingham, UK, 2016.

[32] Ivan Vasilev Daniel Slater Gianmario Spacagna Peter Roelants Valentino Zocca, Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, Packt Publishing ed., 2019.

[33] https://guimperarnau.com/blog/2017/03/Fantastic-GANs-and-where-to-find-them

[34] Joseph Babcock and Raghav Bali, Generative AI with Python and TensorFlow 2, Packt publishing ed. Birmingham, UK, 2021.

[35] https://www.mdpi.com/2624-800X/1/4/37

[36] Éric Biernat et MichelLutz, Data science fondamentaux et etudes de cas machine learning avec python et R, eyrolles ed. Paris/France, 2015.

[37] Connor Shorten, "text-to-image,".

[38] Nikunj Gupta, "Text-to-Image Synthesis," 2019.

[39] https:\\towardatascience.com\text-to-image-a3b201b003ae

[40] Nikunj Gupta, "Text-to-Image Synthesis," 2019.

[41] Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang,Dimitris Metaxas HanZhang, "Text to Image Synthesis Using Generative Adversarial Networks,".

[42] "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks," Rutgers University, Lehigh University, The Chinese University of Hong Kong, Baidu Research,.

[43] Pengchuan Zhang, Qiuyuan Huang, Han Zhang,Zhe Gan, Xiaolei Huang, and Xiaodong He Tao Xu, "Attngan: Finegrained text to image generation with attentional generative adversial network," in conference on computer vision and pattern recognition, 2018, p. 1316–1324.

[44] Jing Zhang, Duanqing Xu, and Dacheng Tao. Tingting Qiao, "Learning text-to-image generation by redescription ," in In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, p. 1505–1514.

[45] Bin Liu, Lu Sheng, Nenghai Yu, Xiaogang Wang, and Jing Shao. Guojun Yin, "Semantics disentangling for text-to image generation.," in In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2327-2336.

[46] Ming Tao1 Hao Tang et Fei Wu et Xiaoyuan Jing et Bing-Kun Bao et Changsheng Xu, "DF-GAN:AS imple and Effective Baseline forText-to-ImageSynthesis," 1Nanjing University of Posts and Telecommunications 2CVL, ETH Z¨urich 3Wuhan University 4Peng Cheng Laboratory 5University of Chinese Academy of Sciences 6NLPR, Institute of Automation, CAS , 2022.

[47] Zhenxing Zhang and Lambert Schomaker, "Dual attention generative adversarial networks for text-to-image," in International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1-8.

[48] Ian Goodfellow, Dimitris Metaxas, and Augustus Odena Han Zhang, "Self-attention generative adversarial networks. ," in International conference on machine learning, 2019, p. 7354–7363.

[49] Andreas Geiger, and Sebastian Nowozin Lars Mescheder, "Which training methods for gans do actually converge," in In International Conference on Machine Learning, 2018, p. 3481–3490.

[50] Mike Schuster and Kuldip K Paliwal., "Bidirectional recurrent neural networks. ," IEEE transactions on Signal Processing, vol. 45, no. 11, p. 2673–2681, 1997.

[51] Ayush Thakur. https://wandb.ai/ayush-thakur/gan-evaluation/reports/How-to-Evaluate-GANs-using-Frechet-Inception-Distance-FID

[52] Jing Zhang, Duanqing Xu, and Dacheng Tao. Tingting Qiao, "Mirrorgan: Learning text-to-image generation by redescription. ," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, p. 1505–1514, 2019.

[53] Bin Liu, Lu Sheng, Nenghai Yu, Xiaogang Wang, and Jing Shao. Guojun Yin, "Semantics disentangling for text-to image generation," Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition, no. 1,2,6,7,8, p. 2327–2336, 2019.

[54] Pingbo Pan, Wei Chen, and Yi Yang. Minfeng Zhu, "Dm-gan:Dynamic Memory generative adversarial networks for text to-image synthesis.," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, no. 1,2,6,7,8, pp. 5802-5810, 2019.

[55] Guosheng Lin, Steven C. H. Hoi Hao Wang, "3D Cartoon Face Generation with Controllable Expressions from a Single GAN Image," JOURNAL OF LATEX, vol. 14, no. 8, AUGUST 2022.

[56] S. Laine, and T. Aila T. Karras, "A style-based generator architecture for generative adversarial network," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognitio, 2019, p. 4401–4410.

[57] G. Lin, S. C. Hoi, and C. Miao H. Wang, "Cycle-consistent inverse gan for text-to-image synthesis," in Proceedings of the 29th ACM international Conference on Multimedia, 2021, p. 630–638.

[58] S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, T. Karras, "Analyzing and improving the image quality of StyleGAN," Proc.CVPR, 2020.

[59] Kevin Chen Christopher B. Choy Manolis Savva Angel X. Chang Thomas Funkhouser Silvio Savarese, "Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings," Stanford University Princeton University, 2018.

[60] *Mordvintsev,Cremers Haeusser*, "*Learning. arXivpreprintarXiv*," 2017.

[61] *H.O.,Xiang,Y., Jegelka,S.,Savarese,S. Song,* "*Deepmetriclearning vialiftedstructuredfeature embedding,*" *Computer Vision and Pattern Recognition,* 2016.

[62] tao X U,hong shēng l i,S hǎotīng Zhang,x AO gāng Wang,x ào léi Huang, Dimitri shì meta xiǎo A shuō Han Zhang, "Text to image synthesis using generative adversarial networks,".

[63] Rajat Garg, "TexttoPhoto-RealisticImageSynthesis," 2019.

[64] *Text 2 Shape :Generating Shapes from Natural Language by Learning*

$JointEmbeddings$. [Online]. $https://arxiv.org/abs/$1803.08495

[65] $A.X.,Funkhouser,T.,Guibas,L.,Hanrahan,P.,Huang,Q.,Li,$ $Z.,Savarese,S.,Savva,M.,Song,S.,Su,H.,Xiao,$ $J.,Yi,L.,Yu,F$ $Chang,$ "$ShapeNet$: An $information$ $-$ $rich$ $3D$ $mode$ $lrepository.TechnicalReportarXiv$: 1512.03012," $StanfordUniversity^{-}$,.

[66] Vishnu Subramanian, Deep Learning with PyTorch. Birmingham, UK: PACKT, 2018.

[67] https://medium.com/deep-learning-demystified/deep-nlp-word-vectors-with-word2vec-d62cb29b40b3

[68] Vishnu Subramanian, Deep Learning with PyTorch , Packt ed. BIRMINGHAM , UK: Packt Publishing, 2018.

[69] Zisserman A Wiles O, "Single-andMulti-View Reconstruction by Learning from Silhouettes.," in BritishMachineVisionConference, 2017.

[70] h$ttps$: $//developpaper.com/transfer$ $-$ $learning$ $-$ nlp $-$ $visual$ $-$ $diagrams$ $-of$ $-bert$ $-elmo$ $-etc/$

[71] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[72] https://shapenet.org/