LIFPROJET - Classification et génération d'images

Ines TOUIL
Youssef ALBAGOURY
Gernido HANAMPATRA

Décembre 2022

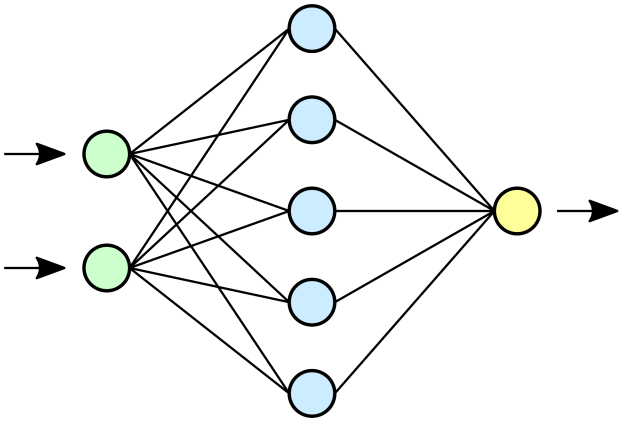

Figure 1: Réseau de Neurones

Contents

1 Introduction

La classification et la génération d'images sont des domaines d'étude importants en informatique et en apprentissage automatique. Ces techniques sont de plus en plus utilisées dans divers domaines, notamment la santé, la sécurité, l'industrie automobile, ou le divertissement.

Les récentes avancées dans le domaine du deep learning ont conduit au développement de techniques plus efficaces, telles que les réseaux neuronaux convolutifs (CNN) et les réseaux adversariens génératifs (GAN). Ces nouvelles approches se sont avérées très efficaces dans un large éventail de tâches, notamment la classification d'images, la détection d'objets et la génération d'images.

L'un des principaux avantages de l'utilisation des réseaux CNN et GAN pour la classification et la génération d'images est leur capacité à apprendre automatiquement des caractéristiques complexes à partir de données brutes. Cela permet d'obtenir des modèles plus précis et plus flexibles, capables de s'adapter à de nouvelles données et tâches. En outre, les CNN et les GAN sont capables de traiter efficacement de grandes quantités de données, ce qui les rend bien adaptés à ce genre de tâches.

Par conséquent, ces techniques sont devenues de plus en plus importantes dans un large éventail d'applications et devraient continuer à jouer un rôle clé dans le domaine de l'informatique et de l'apprentissage automatique.

2 Travail effectué

2.1 Computer vision

Nous avons du apprendre des notions de computer vision. Nous avons compris des concepts tels que :

- La nécéssite d'avoir des descripteurs multiéchelles invariants par certaines transformations géometriques
- L'usage des noyaux de convolutions pour réaliser l'extraction des descripteurs est très commun
- La plupart des tâches de computer vision sont désormais gérées par des réseaux de neurones : il s'agit d'un changement de paradigme puisque les descripteurs sont appris via une backpropagation jusque dans les poids des noyaux, au lieu d'être modélisés comme cela se faisait avant (Méthode SIFt par exemple.)

Cet apprentissage est passé par la consultation de plusieurs sources, que l'on reporte dans la dernière section de ce rapport.

2.2 Réseaux neuronaux

2.2.1 Travail préalable

Nous avons d'abord jugé bon de coder un réseau de neurones de façon assez manuelle pour nous approprier le fonctionnement qui le régit. Nous avons donc codé un Dense Neural Network "from scratch", que nous avons entraîné sur la base MNIST en suivant un tutoriel.

2.2.2 CNN

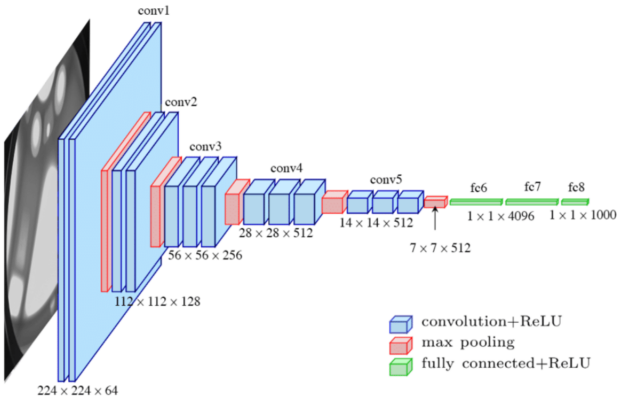

Figure 2: VGG16 CNN Architecture du CNN VGG

Un CNN (Convolutional Neural Network) est un type de modèle d'apprentissage automatique utilisé pour l'analyse d'images. Il utilise des filtres convolutionnels pour extraire des caractéristiques des images, puis les utilise pour effectuer une tâche spécifique, comme la classification d'images. Un CNN contient plusieurs couches de neurones, chacune d'entre elles utilisant des filtres convolutionnels de taille différente pour extraire des caractéristiques à différents niveaux de détails dans l'image. Le résultat final est un modèle capable de reconnaître des objets et des motifs dans les images avec une précision élevée.

Les CNN utilisent des techniques de régularisation pour éviter l'overfitting, c'est-à-dire pour éviter que le modèle ne soit trop spécialisé sur les données d'entraînement et ne soit pas capable de généraliser des résultats sur des données inconnues.

- Pour mettre en place ce projet, j'ai dû apprendre plusieurs concepts théoriques clés en matière de machine learning et de réseaux de neurones convolutionnels (CNN). La théorie de la classification en machine learning m'a permis de comprendre comment entraîner des modèles pour prédire des classes à partir de données d'entraînement étiqueté. J'ai également appris à utiliser des réseaux de neurones simples, qui sont une forme de modèle de machine learning basé sur une série de couches de neurones interconnectées. Enfin, j'ai étudié l'architecture des CNN dont j'ai fait l'explication sommaire plus haut.

- Pour mettre en pratique mes connaissances et implémenter mon modèle, j'ai utilisé Python avec Tensorflow et Keras utilisés ensemble pour créer rapidement des modèles. J'ai également pu exécuter mon code et visualiser les résultats de manière facile et intuitive sur Google Colab.

Pour plus de détails concernant l'implémentation : *https://colab.research.google.com/drive/1lbkajIdUV5-40hT6nLI2qKgwnpi-wTlf?usp=sharing*

2.2.3 GAN

Un GAN (Generative Adversarial Network) est un modèle d'apprentissage automatique qui utilise un système de deux réseaux de neurones entraînés en concurrence. Le premier réseau, appelé générateur, produit des données qui ressemblent à des données réelles, tandis que le second réseau, appelé discriminant, essaie de les différencier. Les deux réseaux s'entraînent ensemble pour produire des résultats de haute qualité. Le résultat final est un générateur capable de produire des données convaincantes qui ressemblent à des données réelles. (Pour plus de détails, cf: https://www.kaggle.com/code/youssefalbagoury/gan-with-keras/notebook)

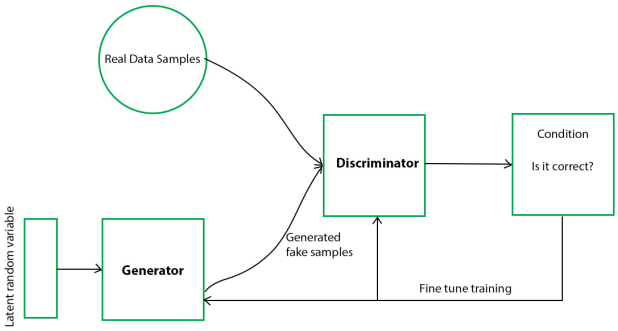

Figure 3: Illustration du fonctionnement d'un GAN. Source : Geeks for Geeks.

2.2.4 Diffusion

Les modèles de diffusion sont des modèles génératifs : ils sont utilisés pour générer des données similaires aux données sur lesquelles ils sont entraînés. Essentiellement, les modèles de diffusion fonctionnent en détruisant les images par l'ajout successif de bruit gaussien, et apprennent ensuite à revenir à l'état initial de l'image en inversant le processus de bruitage. Après l'entraînement, nous pouvons utiliser le modèle de diffusion pour générer des images en faisant passer du bruit échantillonné au hasard dans le processus de débruitage appris, jusqu'à avoir une image sans bruit. Ils sont intialement constitués de 3 éléments :

- **Noise Scheduler** : Fonction qui permet d'ajouter du bruit à une image.
- **Timestep Encoding** : Une fonction qui encode le timestep dans laquelle se trouve l'image (qui est plus ou moins bruitée en fonction de son timestep).
- **U-Net** : Un réseau de neurones à convolution en forme de U : Lors de la première partie (descendante), la taille de l'image est fortement réduite, mais sa profondeur augmente du fait de l'application de filtres de convolutions de plus en plus nombreux. Ensuite, lors de la seconde phase (montante), la taille de l'image est augmentée jusqu'à atteindre sa taille initiale. Ainsi, lorsque l'on fait passer une image dans un U-Net, l'image en sortie a les mêmes dimensions que celle en entrée, ce qui permet de générer des images de hautes résolutions, entre autres. Elle prend en entrée une image bruitée avec un certain timestep, et apprend à prédire la densité de probabilité du bruit de l'image au timestep précédent.

Pour plus de détails concernant l'implémentation : *https://www.kaggle.com/gaahan/diffusion*

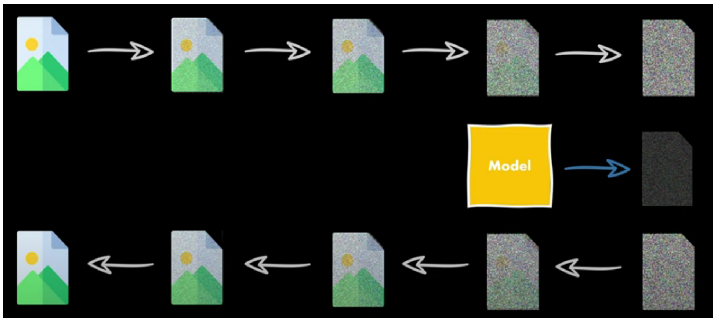

Figure 4: Vue d'ensemble d'un modèle de Diffusion.

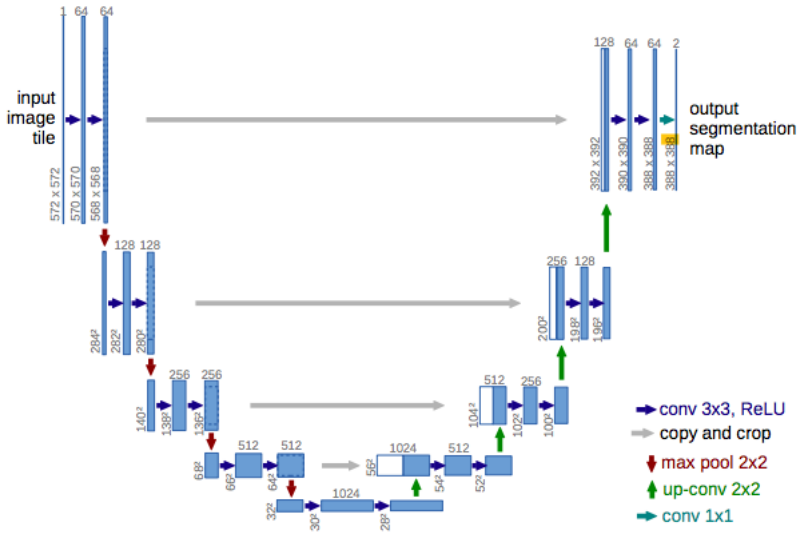

Figure 5: U-Net Architecture.

2.3 Application/Site web

Déploiement avec Flask pour la partie serveur en python - afin d'importer le modèle, de traiter les données du dessin sur le canvas, et de classifier ledit dessin en temps réel. La partie dessin est gérée par des fonctions en javascript, en manipulant le canvas.

3 Organisation du groupe

3.1 Outils

Google Colab, GitForge, Kaggle

3.2 Gestion opérationnelle

Fréquence des réunions, partage colab/application, création d'un environnement conda

Répartition effective des tâches

- **Inès TOUIL** Formatage des données, choix des architectures de CNN et GANs, mise en place de la pipeline de traitement, entraînement et supervision des réseaux de neurones, mise en place de l'application (HTML/CSS).
- **Youssef ALBAGOURY** Choix du framework du site web, implémentation de zéro d'un réseau de neurones simple entraîné sur le dataset MNIST, génération d'image GAN.
- **Gernido HANAMPATRA** Création du site web, et des fonctionnalités associées. Mise en place de la pipeline de traitement. Travail sur le modèle de Diffusion pour la génération d'image : formatage des données, création du modèle (Noise scheduler, Timestep Encoding, U-Net) puis entraînement.

3.3 Décisions importantes

- Ajouter une fonctionnalité de génération d'images : gros défi, réalisé que partiellement faute de temps et disponibilité de certains membres

- Choisir entre tensorflow et pytorch (pour le CNN) : tensorflow est un peu plus simple que pytorch pour une première expérience, assez utilisé dans l'industrie
- Décider de créer une interface utilisateur via un site : un défi supplémentaire pour pouvoir intéragir avec notre travail et diversifier les compétences employées (ML et développement d'application)
- Choix de rédiger le rapport en LaTeX plutôt que Word ou autre outil de traitement de texte pour avoir une première expérience avec un outil standard utilisé par les scientifiques.
- Choix du framework du site web : NuxtJS ou HTML/Javascript. Le choix initial (NuxtJS) a entraîné une refonte complète du site par la suite.

4 Obstacles

- **Le resizing et le formatage des données:** Un mauvais formatage des données a entraîné des erreurs de prédiction sur le site web.
- **Comprendre les réseaux de neurones :** Objet complexe, faisant appel à un framework greffé sur un langage que l'on ne connaissait pas jusqu'ici, il y avait donc beaucoup de choses à assimiler, apprendre à débugger. Certains concepts pourtant simples ne sont pas facile à comprendre si l'on ne s'intéresse pas aux fondements théoriques (backpropagation et descente de gradient).
- **Mise en place du venv de l'application :** Tentative de création d'un venv pour y installer les librairies requises, afin que le site soit utilisable sans avoir à installer de dépendances du côté utilisateur. Toutefois, étant sous Windows, cela n'a pas fonctionné (plusieurs erreurs rencontrées).

5 Bilan

5.1 Compétences acquises

Il s'agit d'un premier contact avec

- Pip, Conda, Python et Flask. Nous avons pu mettre en place un site web dynamique fonctionnant sous Flask : l'utilisateur peut intéragir avec lui en dessinant, puis classifier son dessin.
- Les modèles de classification et génération d'images : CNN, GAN, Diffusion.
- Tensorflow pour créer un modèle de CNN, afin de classifier des images. Pytorch pour créer un modèle de Diffusion et manipuler des tenseurs.

On a également pu

- Gagner en expérience en terme de gestion de projet de bout en bout.
- Affiner notre compréhension du Machine Learning, notamment sur des concepts comme le sur-apprentissage. Nous avons pu mettre en évidence celui-ci lorsque nous n'appliquions pas de dropout ou de calibration des poids du réseau sur un ensemble de validation.

5.2 Piste d'améliorations

- Le CNN a été entraîné pour reconnaître 100 classes différentes. Or le dataset de Quickdraw a 345 classes d'objets différentes. Une amélioration possible serait donc d'entraîner le modèle à reconnaître plus de classes.
- Concernant le site web, lorsque l'on veut uploader une image il faut appuyer une seconde fois sur le bouton pour uploader l'image. Normalement, l'image devrait se mettre automatiquement dans le canvas. Cette partie est donc améliorable.

- Nous n'avons pas eu assez de temps pour terminer la génération d'images. Notamment en ce qui concerne la Diffusion, le modèle ne parvient pas à générer des images (il ne génère que du bruit pendant la phase d'entraînement). Cela peut être dû à plusieurs éléments :

  1) Le modèle n'a pu être entraîné que sur 2 epochs sur 100.
  2) Pour une raison inconnue, les informations de l'image sont entièrement détruites à partir de 30 images sur 300, par ajout de bruit défini dans le Noise Scheduler. Pour résoudre ce problème, une tentative serait de fortement baisser la valeur de β_max (cf: https://www.kaggle.com/gaahan/diffusion).

  Toutefois, si l'on met cela de côté, le réseau U-Net semble bien apprendre (le coût diminue beaucoup entre l'epoch 0 et l'epoch 1).

6 Bibliographie

[**1**] Ian Goodfellow & al., Generative Adversarial Networks, 2014

[**2**] Stanford University, CNN Architectures, cours sur Youtube, 2017

[**3**] CS231n, Deep Learning for Computer Vision Stanford, notes de cours en ligne, 2022

[**4**] Dana H. Ballard, Christopher M. Brown, Computer Vision, 1982

[**5**] DeepFindr, Diffusion models from scratch in PyTorch, Youtube, 2022

[**6**] Zaid Alyafeai, Train a model in tf.keras with Colab, and run it in the browser with TensorFlow.js, Medium, 2018