Table des matières

Liste des tableaux

Liste des figures

Liste d'Acronymes et Abréviations

DA : Domain adaptation.

ITIT : Image to image translation.

DL : Deep learning.

ML : Machine learning.

AI : Intelligence artificielle.

NNs : Réseaux de neurones.

CNN : Réseau de neurones convolutifs.

RNN : Réseau de neurones récurrents.

AE : Auto-encodeur.

CV : Computer vision.

OCR : Optical character recognition.

TALN : Traitement automatique de la langue naturelle.

UNIT : Unsupervised Image-to-Image Translation.

MUNIT : Multimodal Unsupervised Image-to-Image Translation.

Introduction générale

L'extraction des connaissances est la création des connaissances à partir des sources structurées comme des bases de données, *etc*, ou à partir des sources non structurées comme des images, des document PDF, *etc*. Les connaissances qui en résultent doivent être lisibles et interprétables par la machine. Une des méthode d'extraction du texte à partir des images est la reconnaissance optique des caractères (OCR), L'OCR est l'utilisation de l'apprentissage automatique pour reconnaitre et extraire les caractères de texte imprimés ou manuscrits dans les images numériques de documents physiques, tels que les documents papier scannés.

Le processus de base de l'OCR consiste à examiner le texte d'un document et traduire les caractères en format lisible par la machine qui peut être utilisé pour le traitement des données. L'OCR est parfois aussi appelée reconnaissance de texte. Lorsqu'un caractère est identifié, il est converti en un code ASCII qui peut être utilisé par les systèmes informatiques pour d'autres manipulations ou pour le stockage tout simplement.

La reconnaissance du texte manuscrit est la partie la plus complexe d'OCR, cette tâche fait appel à plusieurs disciplines tel que la reconnaissance de forme, traitement automatique du langage naturel, traitement d'image *etc*. L'une d'utilisation d'OCR est dans le monde des banques, particulièrement, l'automatisation de l'extraction des informations des chèques, pour le stockage ou pour n'importe quelles manipulations. L'extraction a presque été parfaite pour les chèques imprimés, mais le problème se pose lors du traitement des chèques manuscrits, cette technologie donne des résultats moins précis, ce qui nécessite une intervention pour faire une révision manuelle de l'information extraite.

Ce travail propose une méthode d'extraction d'une information manuscrite à partir d'un chèque, c'est le montant. Cette méthode propose de faire une ''traduction'' de l'image du montant manuscrit vers une image du même montant imprimé, pour faciliter la reconnaissance, cela veut dire générer une image du montant imprimé à partir de l'image originale du montant manuscrit, ce qui nous a fait penser aux modèles génératifs.

On a donc ciblé le monde des modèles génératifs, particulièrement les modèles qui font la traduction des images d'un domaine à un autre, cela revient à faire de l'adaptation du domaine ''*domain adaptation*'' c'est un cas particulier du transfert de l'apprentissage ''*transfer learning*'', ou on fait adapté les images du domaine du test (montants manuscrits) à celles du domaine d'apprentissage (montants imprimé). L'idée est donc de convertir une image d'un montant manuscrit en une image d'un montant imprimé qui est plus facile à reconnaitre.

Ce rapport est organisé comme suit : Le premier chapitre est dédié au cadre général du projet, l'organisme d'accueil, la problématique et la solution. Le deuxième chapitre est celui de l'état de l'art, ou on parle de transfert de connaissance et quelques applications, puis, une partie a été consacré pour l'apprentissage approfondi, et ses différentes architectures. Le

troisième chapitre parle de quelques notions de la vision par ordinateur qui sont liées à notre sujet, comme les réseaux génératifs et leur utilisation dans la vision par ordinateur, ainsi que l'approche du transfert d'apprentissage implémenté dans de travail. Le quatrième chapitre présente les expérimentations et les résultats.

Chapitre 1 : Cadre général du projet

Introduction

Dans ce chapitre, nous proposons de situer le projet dans son contexte général, la première partie présente l'entreprise d'accueil, ses services, ses équipes, et ses solutions, ainsi son domaine d'activités. Ensuite, on essayera de présenter la problématique et la solution proposée avec la démarche suivie.

1 L'organisme d'accueil

INDATACORE est une entreprise FINTECH (finance et technologie) qui utilise la technologie pour présenter des solutions qui rend les services financiers et bancaires plus faciles. Cette entreprise propose des solutions et des applications grâce à ces équipes de recherche et développement (R&D) dans les champs Big Data, Data Science et Business Intelligence (BI), ses équipes sont composées des experts consultants, des Data Scientists, Data Engineers et Chercheurs, ils développent des solutions qui sont en premier des modèles prédictives et descriptives pour leurs clients, INDATACORE construit des modèles de reconnaissance faciale, vérification des signatures, validations des documents, etc. Le domaine d'activité de INDATACORE est le secteur de banques ainsi que le secteur de Telecom.

1.1 Fiche descriptive

Nom	INDATACORE
LOGO	INDATACORE Big Data Analytics & Data Science
Secteur d'activité	Banques/télécom
Forme juridique	SARL
Adresse	Technopark, route de Nouacer, CASABLANCA.
Contact	+212 520 44 76 40 | contact@indatacore.com
Site Web	www.indatacore.com

Tableau 1 : fiche descriptive de l'entreprise d'accueil.

1.2 Les équipes de travail

INDATACORE est composé de plusieurs équipes qui travaillent sur différentes tâches dans différents champs :

- **L'équipe Big data** : composée de plusieurs profils (Data Architect, Data Fonctionnel, Data Engineers et Data Scientists) qui peuvent intervenir sur plusieurs champs :

  - Constitution du DATALAB et Gouvernance de données.
  - Définition de la stratégie technologique.
  - Architecture et Dimensionnement.
  - Installation et mise en place de l'écosystème Big Data.
  - Tests de performances de l'écosystème Big Data.
  - Identification de différentes sources de données.
  - Branchement de différentes sources de données avec l'écosystème Big Data.
  - Nettoyage et Traitement de la donnée.
  - Préparation de la donnée afin d'extraire la valeur.
  - Etc.

- **L'équipe business-intelligence :** les activités de cette équipe composée des consultants experts sont surtout destinées aux secteurs Banking et télécom, on peut citer :
  - Scoring client.
  - Connaissance client.
  - Rentabilité client.
  - Comportement client.
  - Customer Expérience Management (CEM).
  - Etc.

- **L'équipe data science :** c'est l'équipe dont je faisais partie, cette équipe est composé des data scientistes, des développeurs et des chercheurs, qui interagisse pour résoudre des problèmes dont les solutions sont basées sur :
  - Analyse de données.
  - Analyse de Texte.
  - Traitement d'image.
  - Machine Learning et l'Intelligence Artificielle (AI).
  - Etc.

1.3 Projets et réalisations

Parmi les projets réalisés, on cite quelques-uns qui sont basés sur l'intelligence artificielle et l'apprentissage automatique :

- **SKY Identification** : une solution d'authentifications biométrique basé sur la reconnaissance faciale avec une précision de 99.9%
- **SKY Bank Check** : une solution qui vérifie l'authenticité des signatures, vérification des champs obligatoires dans un chèque, extraction des données d'un chèque (nom, prénom, le code CMC7, la signature...).
- **Sky Document Analysis et Sky Document Detection** : une solution qui applique des techniques de vision par ordinateur dans laquelle les documents sont détectés, isolés et aligné, ces documents extraits sont classés par type (CIN, passport, etc.), les données sont ensuite extraites de ces documents (OCR).

2 Problématique

L'extraction de texte imprimé à partir des images est parmi les tâches complexes de la vision par ordinateur, mais les caractères écrits à la main font passer les choses à un autre niveau de complexité. Contrairement aux polices standards, les textes écrits à la main contiennent rarement des modèles réguliers ou prévisibles (c'est que les ordinateurs cherchent lorsque on leur demande de trouver du texte dans une image). On parle de *l'Optical Character Recognition* (OCR), ou la reconnaissance optique de caractères, c'est la technologie qui permet aux logiciels de reconnaître le texte dans une image. En fait, la capacité de l'OCR à extraire du texte à partir de graphiques ou de documents en fait un outil extrêmement utile dans plusieurs champs, le monde des banques en fait partie, l'OCR est utilisée pour traiter les chèques sans intervention humaine, un chèque peut être inséré dans une machine, l'écriture est numérisée instantanément et le montant exact est transféré. Cette technologie a presque été perfectionnée pour les chèques imprimés, mais le problème se pose pour les chèques manuscrits ou cette technologie donne des résultats moins précis ce qui nécessite une confirmation manuelle.

Le montant dans un chèque est une information très sensible, elle devrait être bien lue par l'ordinateur, pendant ce stage, notre mission était de traiter spécifiquement le champ du montant du chèque, d'autres membres de l'équipe s'occupe de l'extraction d'autres informations.

Une solution qui a été déjà proposée consiste à segmenter le montant est faire la reconnaissance pour chaque chiffre, les limites de cette méthode est que ces montant ne sont pas toujours facile à segmenter, en raison de chevauchement des chiffres.

3 Solution proposée

La solution proposée a été d'adapter des méthodes existantes pour traduire les images des montants manuscrites en des images des montent imprimées, cela revient à creuser dans le monde de transfert d'apprentissage ou *transfer learning* (TL), et faire une étude sur les différentes méthodes existantes qui permettent de traduire les images des chiffres manuscrites en images des chiffres imprimées.

On a donc fait le tour sur plusieurs méthodes en essayant de les adapter pour résoudre notre problème, ces méthodes qui ont comme but la traduction d'une image en entrée pour générer une autre image différente en sortie, elles sont souvent utilisées pour faire les traductions jour en nuit, été en hiver, images réelles en dessins, etc.

Notre but est d'adapter ces méthodes pour traduire une image d'un montant manuscrit en une image du même montant imprimé, pour ensuite faciliter la classification de ces images générées. La classification directe des images contenant les montants manuscrits (sans générer les images des montant imprimés) est faite aussi pour avoir une comparaison entre les résultats de la classification des deux.

4 Objectif

L'objectif est donc de construire un modèle capable de recevoir une image d'un montant manuscrit puis, faire la traduction de cette image en une image contenant le même montant imprimé, pour faciliter la reconnaissance, finalement, ce modèle doit être intégré dans le modèle générale qui fait la reconnaissance pour tous les champs du chèque pour automatiser le processus d'extraction des informations et puis faciliter la tâche pour les banquiers. Ce modèle doit être performant face aux différentes types d'écritures et différente circonstances de la prise de l'image.

Conclusion

Ce chapitre introductif a comme but de mettre en évidence le contexte général du projet et la démarche adoptée pour sa mise en œuvre, on parlait de l'entreprise d'accueil, et de quelques réalisations et projets, on a ensuite posé la problématique suivie de la solution proposée, dans le prochain chapitre, on parle du TL et quelques applications où il est utilisé, vu que c'est notre domaine dont relève notre problématique de recherche, on consacre une partie pour parler de l'apprentissage approfondi à cause de sa relation directe avec le TL, on verra les définitions des concepts qu'on aura besoin pour comprendre la thématique du problème.

Chapitre 2 : Etat de l'art

Introduction

Dans ce chapitre, on donne une idée générale sur le transfert d'apprentissage, sa comparaison avec l'apprentissage traditionnel, on présente aussi ses catégories, puis, on parle d'adaptation de domaine comme un cas du transfert d'apprentissage, avec quelques exemples d'utilisations. La deuxième partie creuse dans le monde d'apprentissage approfondi comme une notion très liée à l'adaptation de domaine.

I. Transfer learning

1 Introduction

Le transfert d'apprentissage est le fait d'utiliser les connaissances acquises dans une tâche pour améliorer l'apprentissage dans une autre tâche connexe. En tant qu'être humain, on a la capacité et les moyens de transférer les connaissances entre différentes tâches, ce qui nous permet d'appliquer cette connaissance tirée des expériences antérieurs lorsque nous rencontrons de nouvelles tâches. Plus une nouvelle tâche est liée à notre expérience antérieure, plus nous pouvons la maîtriser facilement.

Contrairement aux algorithmes traditionnels d'apprentissage automatique qui traitent des tâches isolées, le transfert d'apprentissage tend à être plus flexible, il développe donc des méthodes pour transférer les connaissances acquises dans une ou plusieurs tâches dites « source » et de les utiliser pour améliorer l'apprentissage dans une tâche dite « cible ».

Le transfert d'apprentissage devient utile lorsque nous avons beaucoup de données pour le domaine source et généralement peu de données dans le domaine cible, disons par exemple que nous avons une base de données de millions d'images pour un modèle de reconnaissance d'image, ces données sont massique pour entrainer un modèle et apprendre beaucoup de caractéristique utiles de bas niveau, on veut donc réutiliser cette connaissance pour une autre tâche dont on a pas beaucoup de données, par exemple pour entrainer un modèle de classification des tumeurs ou un modèle de diagnostic radiologique, dans ces cas on se dispose que des centaines d'images, donc entrainer un modèle en se basant juste sur ces données ne donnera pas de dons résultats.

2 L'apprentissage traditionnel vs transfert d'apprentissage

L'apprentissage traditionnel reste insuffisant dans plusieurs cas, il est isolé et il traite uniquement des tâches spécifiques, des jeux de données distincts mènent à des modèles distincts et isolés. Aucune connaissance n'est conservée qui puisse être transférée d'un modèle à un autre. Dans le transfert d'apprentissage, on pout exploiter les connaissances

(caractéristiques, poids, etc.) de modèles déjà formés pour construire de nouveaux modèles et même attaquer des problèmes où on a un manque de données.

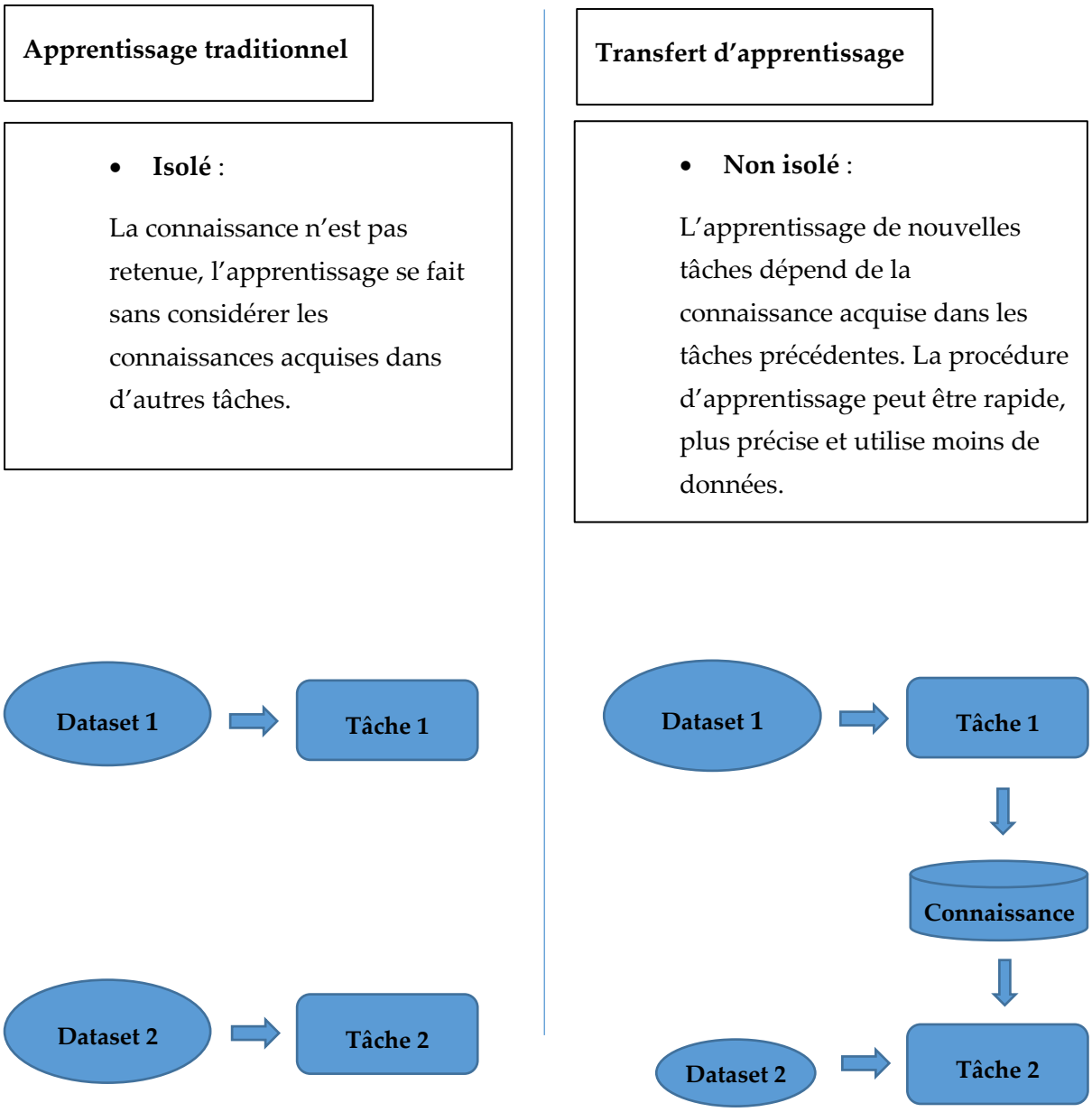


Figure 1 : L'apprentissage traditionnel vs transfert d'apprentissage [1]

3 Notations et définition

Pan et *Yang* dans leur article *A Survey on Transfer Learning* [2] définissent un domaine **D** par deux éléments, un espace de caractéristique χ , et une probabilité marginale **p(X)** avec **X** un échantillon de données. Donc en représente un domaine par : **D = {χ , p(X)}**.

Pour un domaine **D** on peut définir une tâche (*Task*) par deux composants, un espace de classes ou étiquettes **Y** (*Labels*), et une fonction prédictive η qui est appris à partir des données d'entraînement, qui lie les paires **(xi, yi)** avec $xi \in X$ et $yi \in Y$. Pour chaque vecteur dans le domaine **D,** η prédit la classe correspondante : $\eta(xi) = yi$

Donc finalement on peut définir le transfert d'apprentissage comme suit :

Étant donné un domaine source **Ds,** une tâche source **Ts,** ainsi un domaine cible et une tâche cible **Dt** et **Tt** respectivement, le but du transfert d'apprentissage est de permettre de modéliser une distribution de probabilité conditionnelle **P(Yt | Xt)** dans le domaine cible **Dt** en utilisant les informations de **Ds** et **Ts**, avec $Ds \neq Dt$ ou $Ts \neq Tt$.

4 Catégories du transfert d'apprentissage

On peut classer le transfert d'apprentissage selon les différentes situations des domaines et des tâches sources et cibles, si les données de la sources/cibles sont étiquetées ou non, et si les tâches sont les mêmes pour les deux domaines. En se basant sur cela, le transfert d'apprentissage est catégorisé en trois classes : le transfert inductif (*inductive transfer learning*), le transfert transductif (*transductive transfer learning*) et le transfert non supervisé (*unsupervised transfer learning*). Dans [2], les auteurs ont détaillé ces trois catégories.

4.1 Le transfert inductif

Dans le contexte de l'apprentissage par transfert inductif, la tâche cible est différente de la tâche source, peu importe que les domaines source et cible soient identiques ou différents. Dans ce cas, certaines données étiquetées dans le domaine cible sont nécessaires pour construire un modèle prédictif, à utiliser dans le domaine cible. Nous pouvons classer plus en détail l'apprentissage par transfert inductif en deux cas selon la disponibilité des étiquètes dans le domaine source.

4.1.1 Domaine source étiqueté

Dans ce cas, on se dispose de nombreuses données étiquetées dans le domaine source, l'apprentissage par transfert inductif vise dans cette situation à atteindre un haut niveau de performance dans la tâche cible en transférant les connaissances de la tâche source.

4.1.2 Domaine source non étiqueté

Dans ce scénario, Les données étiquetées dans le domaine source ne sont pas disponibles, et les domaines source et cible peuvent être différents, ce qui implique que les informations secondaires du domaine source ne peuvent pas être utilisées directement.

4.2 Le transfert transductif

Dans le cadre de l'apprentissage par transfert transductif, les tâches source et cible sont les mêmes, alors que les domaines source et cible sont différents. Dans cette situation, aucune donnée étiquetée dans le domaine cible n'est disponible alors que de nombreuses données étiquetées dans le domaine source sont disponibles. En plus, et en fonction des situations différentes entre le domaine source et le domaine cible, nous pouvons classer l'apprentissage par transfert transductif en deux cas :

- Les espaces de caractéristiques entre les domaines source et cible sont différents $\chi s \neq \chi t$
- Les espaces de caractéristiques entre les domaines source et cible sont les mêmes $\chi s = \chi t$ mais les distributions de probabilité marginale des données d'entrée sont différentes, $P(Xs) = P(Xt)$

4.3 Le transfert non supervisé

Dans la dernière catégorie, l'apprentissage par transfert non supervisé, semblable à celui de l'apprentissage par transfert inductif, la tâche cible est différente de la tâche source, mais liée à celle-ci. Cependant, l'apprentissage par transfert non supervisé se concentre sur la résolution de tâches d'apprentissage non supervisées dans le domaine cible, telles que le regroupement (*clustering*), la réduction de la dimensionnalité, etc. Dans ce cas, il n'y a pas de données étiquetées disponibles dans les domaines source et cible.

La figure suivante, tirée du document sur l'apprentissage par le transfert mentionné plus tôt, *A Survey on Transfer Learning* [2], résumer toutes les catégories évoquées précédemment.

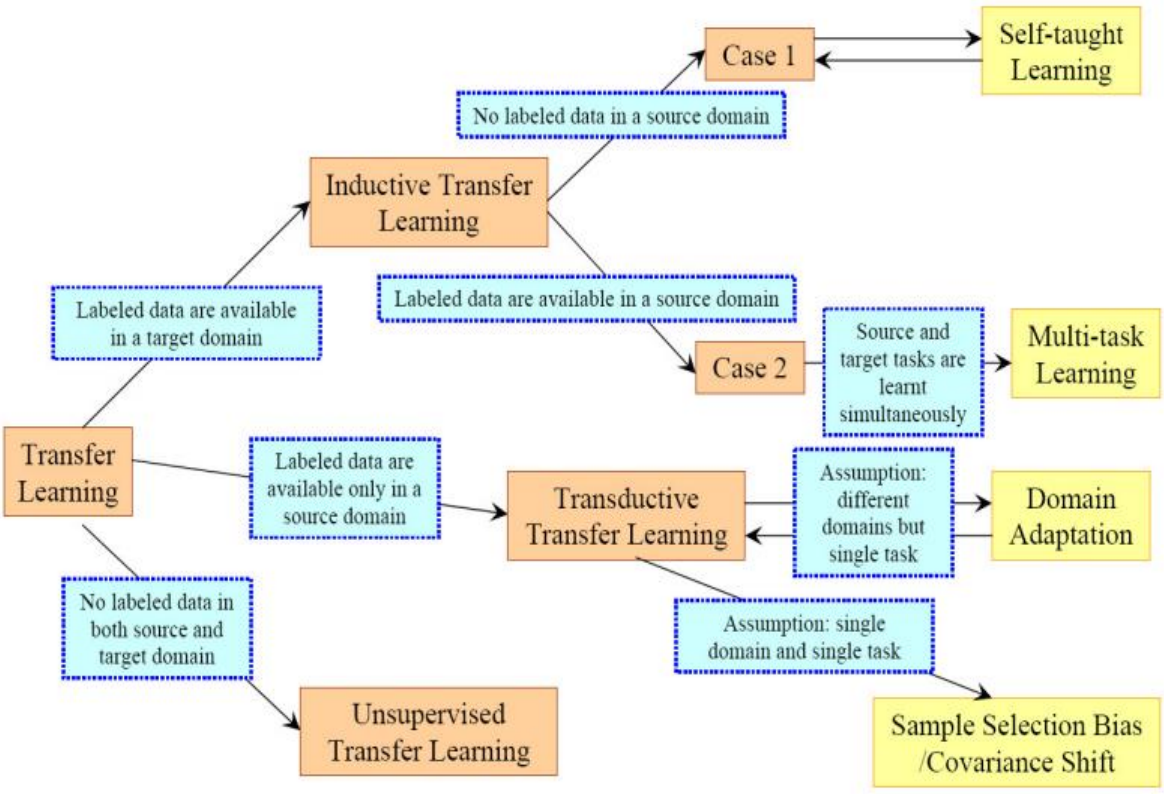


Figure 2 : L'ensemble des différentes catégories d'apprentissage par transfert

5 Applications

Dans cette partie, on va discuter quelques cas d'utilisation du transfert d'apprentissage, tout en résumant quelques articles qui proposent quelques algorithmes et méthodes d'apprentissage par transfert. Il est utilisé dans de nombreux domaines de l'apprentissage automatique, dans différentes tâches comme la classification et le clustering, nous verrons quelques exemples d'applications dans la partie d'adaptation de domaine (*Domain adaptation*) qui est un cas d'apprentissage par transfert.

5.1 Adaptation de domaine DA

On vient de voir que l'adaptation de domaine est un cas d'apprentissage par transfert, pour mieux comprendre, on doit poser la question suivante : quand peut-on dire que deux domaines sont différents ? dans [3], *Wouter* et *Marco* estiment que deux domaines sont différents s'ils sont différents dans au moins une de leurs composantes constitutives, c.-à-d. l'espace de caractéristique χ , l'espace de classes ou étiquettes Y, ou la fonction de densité de probabilité $\mathbf{p(X)}$. L'apprentissage par transfert est défini comme le cas général où les domaines sont librement autorisés à être différents dans les espaces de caractéristiques, de classes, la fonction de densité de probabilité, ou les trois. L'adaptation de domaine est définie comme le

cas particulier où les espaces de χ, et de Y restent inchangés et seulement les distributions de probabilité changent.

5.2 Applications d'adaptation de domaine

Les techniques d'adaptation du domaine ont récemment été appliquées avec succès dans de nombreuses applications du monde réel, notamment la classification d'images, la reconnaissance d'objets, la reconnaissance faciale, la détection d'objets, etc. Dans cette section, nous présentons différents exemples d'application de DA.

5.2.1 Classification d'images

Dans [4], les auteurs ont proposé une méthode qui se base sur l'apprentissage contradictoire ou *Adversarial learning* (AL) c'est une méthode améliorée d'adaptation de domaine non supervisée qui combine AL et l'apprentissage discriminatoire des caractéristiques. Ce modèle proposé fait un mappage d'images de domaine cible à l'espace de caractéristiques sources. On discutera ce type d'apprentissage dans [cf. Chapitre 3] dans la partie des GANs.

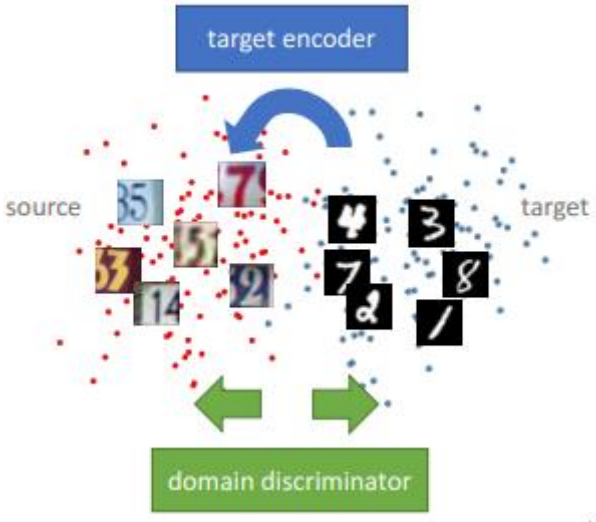


Figure 3 : la méthode de mappage proposé dans [4].

Plusieurs approches AL de l'adaptation non supervisée des domaines ont récemment été introduites, ce qui réduit la différence entre la distribution des domaines d'apprentissage et celle des domaines de test, Ce qui donne des modèles capables pour faire face aux cas plus généraux.

Le Framework proposé dans cet article traite le cas où on a l'accès aux images du domaine source ***Xs,*** et aux étiquettes ***Ys*** tirés d'une distribution de domaine source ***Ps(x, y)***, ainsi que des images du domaine cibles ***Xt.*** L'objectif est d'apprendre une représentation du domaine cible, ***Mt*** et un classificateur ***Ct*** qui permettent de classer correctement les images

cibles dans l'une des ***K*** catégories au moment du test, même si les images cibles ne sont pas étiquetées, ce qui impose la non-faisabilité d'entrainement directe sur ces images. Le DA donc apprend un mappage de représentation de la source ***Ms***, avec un classifieur source ***Cs***, et apprend ensuite à adapter ce modèle pour l'utiliser dans le domaine cible. Le but dans différentes approches du DA est de minimiser la distance entre les distributions de mappage ***Ms(Xs)*** et ***Mt(Xt)***, pour qu'on puisse appliquer le modèle de classification source ***Cs*** directement aux représentations cibles. Dans le cas idéal on aura : **C = Cs = Ct.** [4]

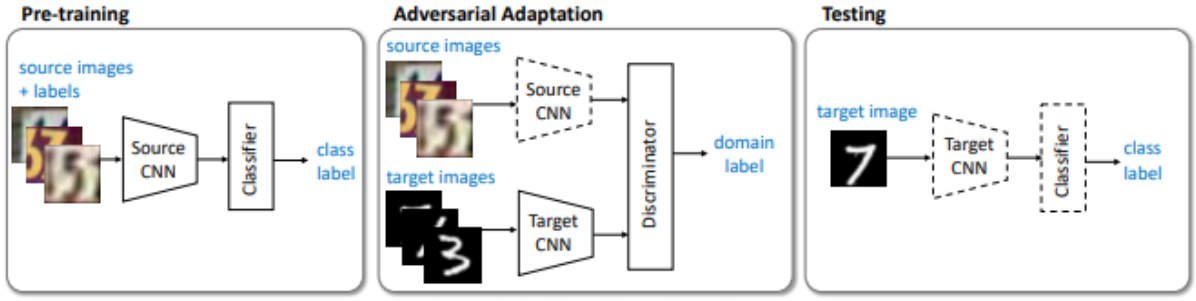


Figure 4 : méthode *Adversarial Discriminative Domain Adaptation* (ADDA) [4]

La figure 4 montre les différentes étapes de cette approche :

- Premièrement ils ont entrainé un modèle qui encode les images source (encodeur source) puis les classifier on se basant sur les étiquettes source.
- Dans la deuxième partie ils ont construit un modèle en performant l'adaptation contradictoire ou AL, le but est d'entrainé un encodeur cible de telle sorte qu'un discriminateur qui voit un exemple codé de source ou de cible ne peut pas prédire de façon fiable l'étiquette de son domaine, autrement dit, le discriminateur ne doit pas pouvoir distinguer entre les images encodées à partir du domaine source et celles encodées du domaine cible.
- Finalement dans la phase de test, les images cibles sont mappées vers l'espace de caractéristiques partagé avec les images source, et donc le classifieur source est utilisé pour prédire leurs classes.

Une autre approche est proposée par Tzeng *et al.* dans [5], ils ont commencé à introduire le problème par un exemple significatif : « Prenons l'exemple d'un groupe de robots formés par le fabricant pour reconnaître des milliers d'objets à l'aide de bases de données d'images standard, puis expédiés aux diffèrent clients partout dans le pays. Comme chaque robot commence à fonctionner dans son propre environnement unique, il est susceptible d'avoir dégradé les performances en raison du changement de domaine. Il est clair qu'avec suffisamment de données supervisées supplémentaires provenant du nouvel environnement,

la performance initiale pourrait être récupérée. » Le problème posé est que le réapprentissage sur le nouvel environnement nécessite un grand nombre d'images étiquetées pour récupérer la performance des modèles, ce qui est presque impossible (dans les cas des robots vendus par exemple), « Il est raisonnable de supposer que le nouveau propriétaire du robot étiquettera quelques exemples pour quelques types d'objets, mais il est totalement irréaliste de supposer une supervision complète dans le nouvel environnement.»

Leur algorithme effectue une adaptation entre l'environnement d'entrainement (source) et l'environnement de test (cible) en utilisant à la fois les statistiques rassemblées à partir des données non étiquetées dans le nouvel environnement, ainsi que les exemples étiquetés manuellement. Le transfert proposé dans cet article ne se fait pas juste au niveau du domaine, mais aussi ou niveau de tâche. Le transfert dans le domaine est accompli en rendant les distributions marginales de caractéristiques de la source et de la cible aussi similaires que possible l'une à l'autre alors que le transfert dans les tâches est effectué par le transfert des corrélations des catégories apprises sur la source vers le domaine cible, Ce qui permet de préserver les relations entre les catégories, cela est la valeur ajoutée dans cette méthode.

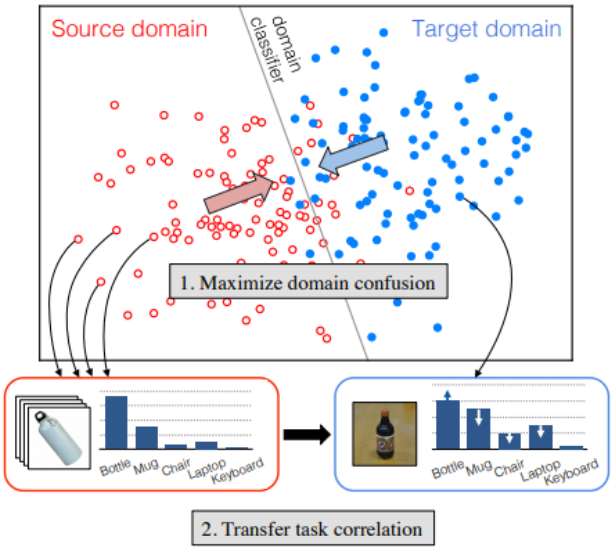


Figure 5 : 1) minimiser la distance entre les distributions du domaine source et du domaine cible en maximisant la confusion de domaine. 2) transfert de la corrélation des classes au niveau de la tâche.

5.2.2 La reconnaissance faciale

Lorsqu'il s'agit de systèmes de reconnaissance faciale, les erreurs ne devraient pas être tolérables dans la plupart des cas, donc la performance des modèles de la reconnaissance faciale devrait être presque parfaite. Dans de nombreuses applications du monde réel ce n'est pas toujours le cas, cette performance se dégrade considérablement lorsqu'il y a des variations

dans les images de test qui ne sont pas présentes dans les images d'entraînement. Pour Réduire l'écart entre les domaines source et cible, Kan *et al.* [6] proposent une méthode d'adaptation de domaine sous le nom de *Bi-shifting Auto-Encoder network* (BAE). Cet Auto-encodeur proposé tente de « déplacer » les échantillons du domaine source vers le domaine cible et de « déplacer » également les échantillons du domaine cible vers le domaine source. Le domaine source décalé est supervisé et suit une distribution similaire à celle du domaine cible, en conséquence, n'importe quelle méthode supervisée peut être appliquée sur le domaine source décalé pour former un classificateur pour la classification dans le domaine cible.

La différence entre un auto-encodeur simple et celui proposé est que le premier essaie de reconstruire l'entrée elle-même, qui est généralement utilisée pour la réduction dimensionnelle, alors que l'auto-encodeur proposé tente de transférer des échantillons d'un domaine à l'autre pour traiter l'écart entre les domaines (Figure 6).

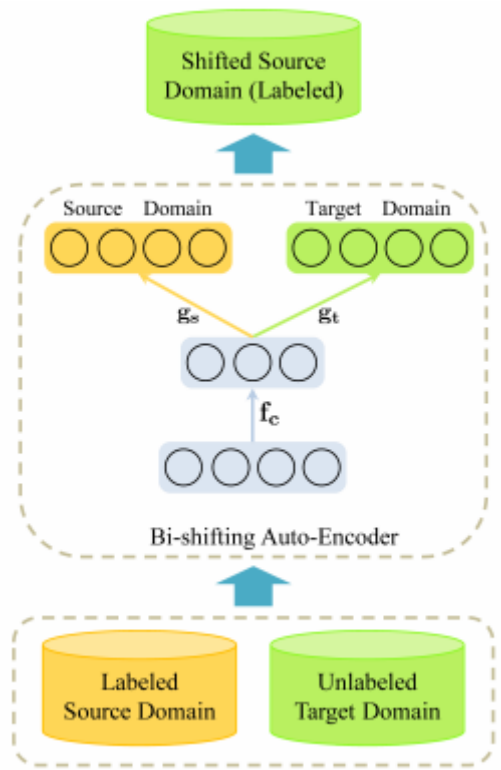


Figure 6 : la structure générale du réseau proposé (BAE) [6].

Une autre utilisation d'adaptation de domaine dans le champ de la reconnaissance faciale a donné des résultats remarquables, celle proposé dans [7] par Hong *et al.*, cette méthode qui utilise seulement un seul visage par personne ou SSPP (*single sample per person*), ce modèle qui est construit en se basant à la fois sur l'adaptation de domaine, l'extraction des caractéristiques et la classification en utilisant une architecture profonde. Pour entrainer un

réseau profond, une seule image par classe reste insuffisante, Pour résoudre ce problème, ils ont généré des images synthétiques (pour chaque image donnée) avec des poses différentes à l'aide d'un modèle de visage 3D, fig. 7 (b).

L'utilité de l'adaptation de domaine dans cette approche est résumée par les auteurs comme suit : « généralement en DA, un mappage entre le domaine source et le domaine cible est effectué, de sorte que le classificateur construit pour le domaine source peut également être appliqué au domaine cible. Inspirés par ceci, nous supposons des conditions de prise de l'image stables pour construire une galerie définie comme domaine source et des conditions instables pour construire une galerie définie comme domaine cible. »

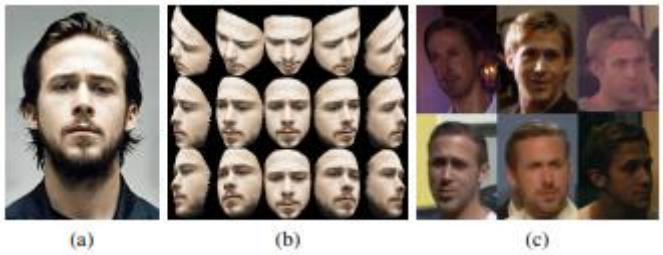


Figure 7 : (a) exemple de la base stable (source), (b) les images synthétiques générées. (c) exemples des images non stables utilisées dans le test (cible)

5.2.3 Image to image translation (ITIT)

ITIT est une classe de la vision par ordinateur dont l'objectif est d'apprendre un mapping entre une image d'entrée et une image de sortie, elle a récemment connu un grand succès avec le DA profond, elle a été appliquée à diverses tâches, telles que le transfert de style l'amélioration des photos, etc. Tout comme l'apprentissage traditionnel, ITIT peut être catégorisé comme supervisé et non supervisé, dans le cas supervisé, les images correspondantes sont appariées dans différents domaines alors que dans le contexte non supervisé, nous n'avons que deux ensembles d'images indépendants, l'un composé d'images d'un domaine et l'autre d'images d'un autre domaine il n'existe aucun exemple apparié montrant comment une image pourrait être traduite en une image correspondante dans l'autre domaine [cf. Chapitre 3].

Dans [8], Isola *et al.* proposent une architecture supervisée, qui utilise un GAN conditionnel pour apprendre un mapping des images sources aux images cibles. Cette architecture nommée *pix2pix* pour *pixels to pixels* définit la traduction automatique d'image en image comme la traduction d'une représentation possible d'une scène en une autre, l'objectif principal de cette méthode est de prédire les pixels de la sortie à partir des pixels donnés dans l'image en entrée.

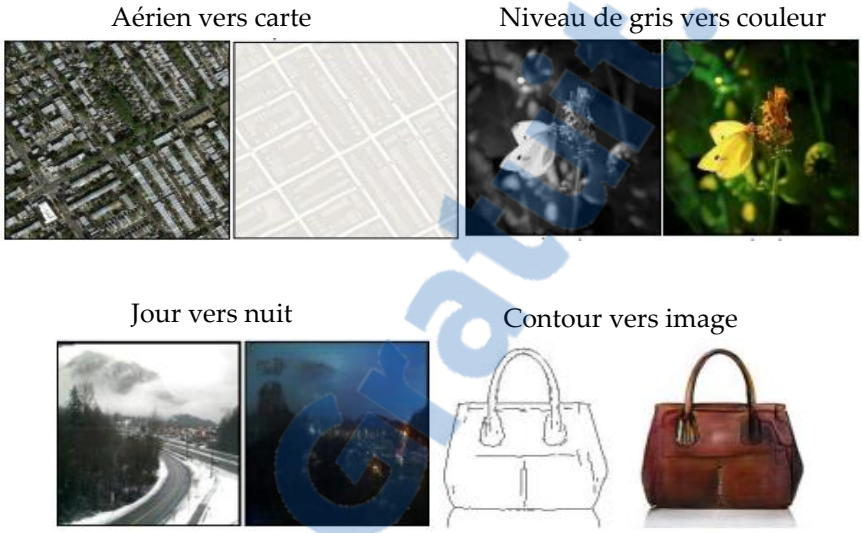


Figure 8 : Quelques exemples de la traduction *pix2pix* présentés dans [8]. Les images à gauche sont celles données en entrée, les images à droite sont celles générées par le réseau proposé.

Une autre approche dans la partie supervisée est celle proposée par Tzeng *et al.* dans [9], cette approche essaye de trouver une solution pour les problèmes de robotique du monde réel qui surviennent souvent dans des domaines qui diffèrent considérablement de l'environnement d'entraînement antérieur des robots. Cette nouvelle approche d'adaptation de domaine pour la perception robotique adapte les représentations visuelles apprises sur un grand ensemble de données source facile à obtenir (par exemple des images synthétiques) à un domaine réel cible, sans nécessiter l'annotation manuelle coûteuse des données du monde réel. Donc en adaptant ces images synthétiques (simulations) qui sont facile à étiqueter, on obtient des images du monde réel avec lesquelles on entraine ces robots.

Les applications sont plus nombreuses dans la catégorie non supervisée, les images des deux domaines ne sont pas appariées, ce qui facilite la tâche au niveau de collection de données, et au niveau du pré traitement. Dans [10] murez *et al.* proposent un *Framework* pour l'adaptation non supervisée du domaine, qui permet de tester des réseaux neuronaux profonds entrainés sur un domaine source sur un autre domaine cible sans avoir besoin des étiquettes d'entrainement dans le domaine cible. Ils entrainent leur modèle de manière à ce

que les caractéristiques extraites par leur encodeur puissent reconstruire les images dans les deux domaines.

II. Deep learning (DL)

1 Introduction

L'apprentissage approfondi en anglais *deep learning* est un sous-domaine de l'apprentissage automatique (*machine learning ML*) qui utilise des algorithmes inspirés par la structure et le fonctionnement du cerveau humain, appelés réseaux de neurones artificiels (NNs). En d'autres termes, il reflète ou il essaye d'imiter le fonctionnement de notre cerveau. Les algorithmes d'apprentissage approfondi sont semblables à la façon dont le système nerveux se structure là où chaque neurone se connecte à un autre, avec l'information qui est transmise de l'un à l'autre. Le mot « approfondi » fait référence à la création de réseaux neuronaux profonds. Il s'agit donc d'un réseau neuronal avec un grand nombre de couches, qui améliore sa capacité à résoudre des problèmes plus compliqués.

« Quand vous entendez le mot *deep*, pensez simplement à un grand réseau neuronal, *deep* fait référence au nombre de couches typiquement et c'est donc le terme populaire qui a été adopté » *Jeff Dean* le leader de Google.ai, la division AI de Google.

Ce domaine a été introduit dans le but de rapprocher l'apprentissage automatique de l'un de ses objectifs initiaux : Intelligence Artificielle (AI). Ce concept n'est pas nouveau, il existe depuis quelques années, ça attire de plus en plus l'attention de nos jours parce qu'auparavant, nous n'avions pas beaucoup de puissance de calcul et beaucoup de données, récemment et au cours des 20 dernières années, la puissance de traitement augmente de façon exponentielle, ce qui a donné l'opportunité à DL et ML pour faire un vrai départ.

L'apprentissage approfondi utilise ce qu'on appelle l'apprentissage supervisé où le réseau neuronal est formé en utilisant des données étiquetées, ou l'apprentissage non supervisé où le réseau utilise des données non étiquetées. Les neurones à chaque niveau font leurs "suppositions" et leurs prédictions les plus probables, puis transmettent cette information au niveau suivant, jusqu'au résultat final.

On comprend que ces réseaux approfondis sont considérés comme une évolution des réseaux de neurones traditionnels, voici donc quelques-unes des facettes de cette évolution :

- Plus de neurones que les réseaux précédents.
- Façons plus complexes pour relier les couches et les neurones dans les NNs.
- Explosion de la puissance de calcul disponible pour entrainer les modèles.
- Extraction automatique des caractéristiques.

2 DL vs ML

Si l'apprentissage automatique est un sous-domaine de l'intelligence artificielle, alors l'apprentissage approfondi pourrait être appelé un sous-domaine de l'apprentissage automatique. La principale différence entre l'apprentissage approfondi et l'apprentissage automatique est que les modèles ML s'améliorent progressivement mais que le modèle a encore besoin d'être guidé. Si un modèle ML renvoie une prédiction incorrecte, le programmeur doit résoudre ce problème explicitement, mais dans le cas de DL, le modèle le fait lui-même. Le système de conduite automatique est un bon exemple de DL.

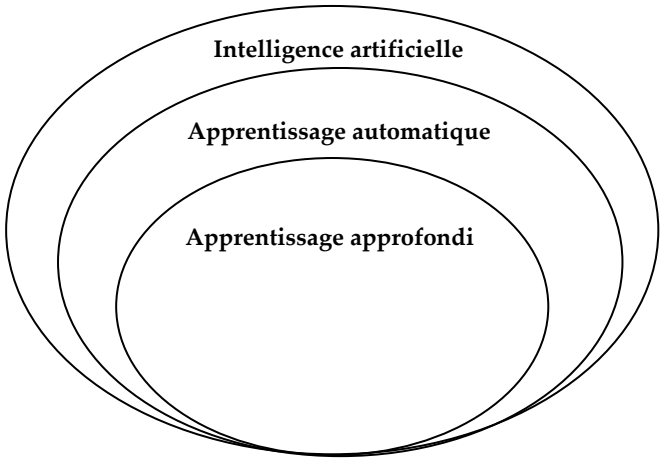


Figure 9 : la relation entre les trois champs AI, ML, et DL.

Le flux des tâches dans ML commence par l'extraction manuelle des caractéristiques pertinentes à partir des images, ces caractéristiques sont ensuite utilisées pour créer un modèle. Grâce à un processus DL, les caractéristiques pertinentes sont automatiquement extraites des images, par extraction de caractéristiques, nous entendons le processus qui consiste à décider quelles caractéristiques d'un ensemble de données peuvent être utilisées comme indicateurs pour étiqueter ces données de façon fiable, avent, les chercheurs dans l'apprentissage automatique ont passé des mois ou même des années, de leur vie à créer manuellement des ensembles de caractéristiques exhaustifs pour la classification des données. Les algorithmes DL ont absorbé des décennies d'efforts humains, car ils avaient accumulé des caractéristiques pertinentes pour classer les données en entrées. [11]

On comprend alors que le DL permet un apprentissage de bout en bout, c'est-à-dire qu'un réseau reçoit des données brutes et une tâche à exécuter, comme la classification, et apprend comment le faire automatiquement. Le tableau suivant résume les points majeurs où DL et ML se diffèrent :

ML	DL
Fonctionne sur une petite quantité de jeux de données	Fonctionne sur une grande quantité de jeux de données.
Ne nécessite pas des machines puissantes	Lourdement dépendant des machines puissante
Divise les tâches en sous-tâches, les résout individuellement et finalement combine les résultats.	Résout les problèmes de bout en bout.
Il faut moins de temps pour s'entraîner.	Il faut plus de temps pour s'entraîner.
l'intervention d'un expert du domaine est nécessaire parfois pour aider dans l'extraction des caractéristiques	Pas obligatoire

Tableau 2 : comparaison entre ML et DL.

3 Architectures DL

Quand on parle de DL, on parle de beaucoup d'architectures qui peuvent être utilisées en fonction de la tâche à accomplir, nous définirons le DL comme des réseaux de neurones avec un grand nombre de paramètres et de couches dans des quatre architectures de réseaux suivantes :

- Réseau de neurones convolutifs ou réseau de neurones à convolution (CNN).
- Réseau de neurones récurrents (RNN).
- Auto-encodeur (AE).
- Réseaux adverses génératifs (GANs).

3.1 CNN

Cette architecture de réseau a été proposée pour la première fois par *Fukushima* [12] en 1988. Il n'a pas été largement utilisé en raison des limites du matériel de calcul pour la formation du réseau, aujourd'hui, c'est l'une des architectures les plus utilisées dans le monde

de DL, spécialement pour les modèles de classification des images, car elle est bien adaptée dans sa structure pour le traitement des matrices 2D et 3D, et elle est efficace pour apprendre et extraire des abstractions de caractéristiques 2D.

Un CNN simple est une séquence de couches, et chaque couche d'un CNN transforme la donnée en entrée en application plusieurs transformations, nous parlons de trois principaux types de couches pour construire des architectures CNN : Couche convolutive, couche de Pooling, couche de correction ReLu et la couche entièrement connectée, parlons de l'architecture générale, les trois premières couches forment la partie d'extraction des caractéristiques, et la dernière couche forme la partie de la classification, sans parler de la couche d'entrée.

3.1.1 Couche convolutive

La couche de convolution est l'élément de base d'un réseau CNN qui effectue la plupart des opérations de calcul lourd. Il prend une entrée, applique un filtre de convolution (*Kernel*), et nous donne une matrice de caractéristiques dites *features map* en sortie. Une couche de convolution reçoit une image normale sous la forme d'une matrice 2D pour les images niveau de gris, ou 3D pour les images RGB dont la largeur et la hauteur sont mesurées par le nombre de pixels le long de ces dimensions, et dont la profondeur est de trois couches, une pour chaque lettre en RGB. Ces couches de profondeur sont appelées canaux, pour chaque pixel d'une image, l'intensité de R, G et B sera exprimée par un nombre, et ce nombre sera un élément de l'une des trois matrices bidimensionnelles, qui forment ensemble le volume d'image. Lorsque les images se déplacent à travers un réseau convolutionnel, on glisse chaque filtre sur la largeur et la hauteur du volume d'entrée (l'image) et on calcule les produits de points (pixels) entre les valeurs du filtre et l'entrée dans n'importe quelle position. En faisant glisser le filtre sur l'image, nous produirons une carte d'activation bidimensionnelle (matrice de caractéristiques).

L'entrée d'une convolution peut être des données brutes ou une sortie de matrice de caractéristiques d'une autre convolution. Cette opération (Figure 10) est souvent interprétée comme un filtre dans lequel on laisse passer que certains types d'informations des données d'entrée, par exemple, un filtre en contour ne laisse passer que les informations du contour d'une image, Intuitivement, le réseau apprendra les filtres qui s'activent lorsqu'ils voient un certain type de caractéristique visuelle. Les premières couches de convolution activent les filtres des caractéristiques d'un niveau bas, comme les contours, alors que dans les couches supérieures du réseau, en trouve des filtres qui s'activent pour des motifs en forme de visage par exemple. Finalement, nous aurons un ensemble complet de filtres dans chaque couche de convolution, et chacun d'eux produira une carte d'activation bidimensionnelle séparée. Nous empilerons ces cartes d'activation le long de la dimension de profondeur et produirons le volume de sortie (vecteur caractéristique de l'image) [13].

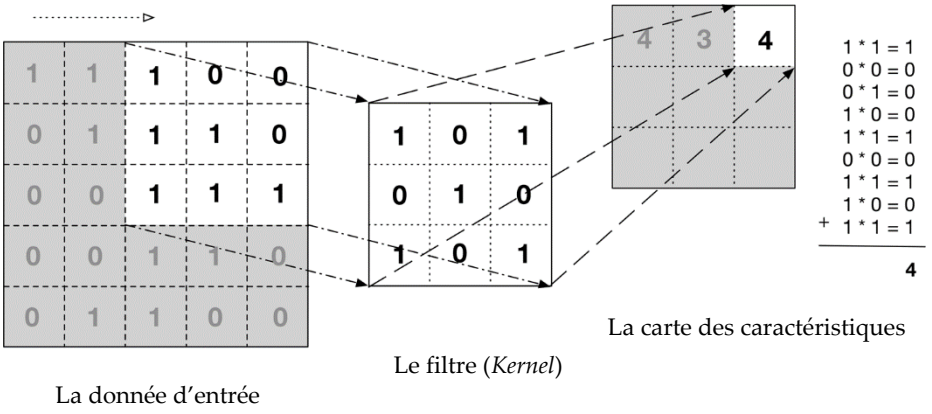


Figure 10 : l'opération de convolution [11].

3.1.2 Couche de Pooling

Les couches de Pooling, aussi connu sous le nom des couches de sous-échantillonnage, sont généralement insérées entre des couches convolutionnelles successives. Nous voulons suivre des couches convolutives avec des couches de Pooling pour réduire progressivement la taille (largeur et hauteur) des matrices représentatives des données. Les cartes de caractéristiques sont introduites dans une couche de sous-échantillonnage, et comme les convolutions, cette méthode est appliquée à une partie de la matrice à la fois (espace de 3x3 par exemple). Dans ce cas, la couche Pooling prend simplement la plus grande valeur d'une partie de la matrice, la mettre dans une nouvelle matrice à côté des valeurs max des autres parties, et rejette le reste des informations contenues dans les cartes de caractéristiques, prenons l'exemple d'un filtre 3x3, La couche de Pooling dans ce cas va utiliser l'opération max() pour redimensionner les matrices d'entrée. L'opération max() prend alors le plus grand des neuf nombres dans la zone de filtrage. Cette opération n'affecte pas la 3eme dimension (profondeur).

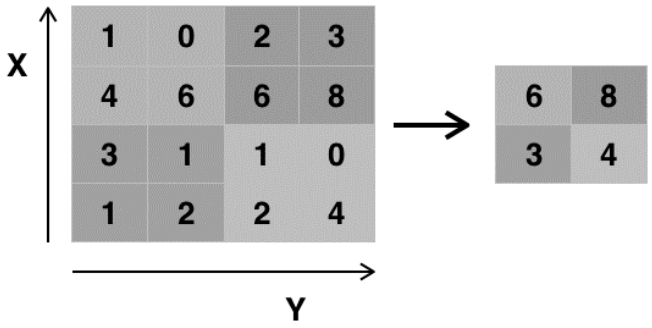


Figure 11 : l'opération de Pooling avec la fonction max(). [15]

3.1.3 Couche de correction ReLU

ReLU est l'abréviation de *Rectified Linear Unit*, Le concept de base de cette couche est simple, garder toutes les valeurs au-dessus de zéro et mettre toutes les valeurs négatives à zéro (Figure 12). Elle applique la fonction d'activation $f(x)=max(0, x)$, elle augmente les propriétés non linéaires de la fonction de décision et de l'ensemble du réseau.

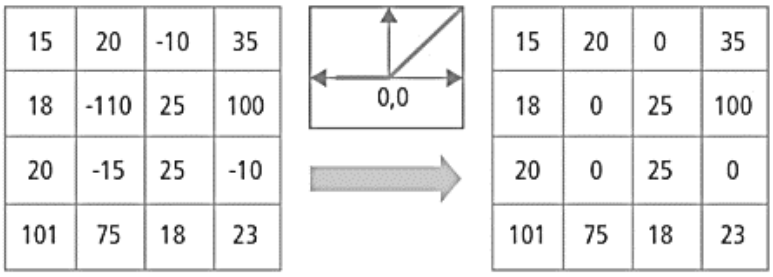


Figure 12 : l'application de la fonction ReLU sur un bloc de 4x4. [16]

3.1.4 Couche entièrement connectée

C'est la dernière couche dans le réseau CNN, après avoir extraire les caractéristiques à travers un long chemin des couches de convolution, Pooling et correction, on passe ces caractéristiques aux couches de classification dans lesquelles nous avons une ou plusieurs couches entièrement connectées pour prendre ces caractéristiques d'ordre supérieur et produire des probabilités de classe ou des scores. Ces couches sont entièrement connectées à tous les neurones de la couche précédente, comme leur nom l'indique. La sortie de ces couches produit typiquement une matrice bidimensionnelle des dimensions [b × N], où b est le nombre d'exemples dans l'entrée, et N est le nombre de classes que nous sommes intéressés à avoir leurs scores.

3.1.5 Résumé

Les réseaux CNN ont évolué en raison de la nécessité d'extraire des caractéristiques spécialisées des données d'image. Nous avons vu comment les couches convolutives travaillent afin de construire des matrices ou cartes caractéristiques, les couches de Pooling réduisent la dimension de ces cartes et les couches régulières entièrement connectées fonctionnaient ensemble pour faire la classification des images. Donc les étapes à suivre lors de la construction d'un CNN sont :

- Fournir des images d'entrées dans la couche de convolution
- Appliquer les filtres de convolution sur les images et appliquer l'activation ReLU à la matrice.
- Effectuer le Pooling pour réduire la taille de la dimensionnalité

- Ajouter autant de couches convolutives que nécessaire jusqu'à ce qu'on obtient un résultat satisfaisant.
- Aplatir la sortie de convolution et l'alimenter à la couche entièrement connectée.
- Donner la classe en sortie à l'aide d'une fonction d'activation.

3.2 RNN

Lorsqu'on est en train de lire un livre ou un article, on comprend chaque mot ou phrase en se basant sur la compréhension des mots ou phrases précédentes. Les NNs ou les CNN n'ont pas la capacité de faire la même chose, la raison pour laquelle ces réseaux ne peuvent pas traiter ce genre de situation est que ces approches ne traitent qu'un vecteur de taille fixe en entrée (une image par exemple) et produisent un vecteur de taille fixe en sortie (les probabilités de différentes classes). Une autre raison est que ces approches fonctionnent en gardant un nombre fixe de couches.

Les RNN sont uniques car ils permettent d'opérer sur une séquence de vecteurs dans le temps. Ils sont différents des autres réseaux feed-forward dans leur capacité à envoyer des informations dans le temps. Les RNN sont utilisés dans la reconnaissance de la parole, la traduction linguistique, et tout ce qui est séquentiel comme des séquences vidéo, etc. Ils apprennent de ce qui s'est passé dans le passé pour mieux prédire l'avenir. Une très bonne explication est présentée par l'un des chercheurs éminents dans le domaine de AI, *Juergen Schmidhuber :* « Les RNN permettent à la fois le calcul parallèle et séquentiel, et peuvent en principe calculer tout ce qu'un ordinateur traditionnel peut calculer. Contrairement aux ordinateurs traditionnels, cependant, les RNN sont semblables au cerveau humain, qui est un vaste réseau de neurones connectés qui, d'une manière ou d'une autre, peuvent apprendre à traduire un flux d'entrée permanent en une séquence de sorties utiles. »

Le mécanisme derrière un RNN est simple, un NN traditionnel se compose d'une couche d'entrée, des couches cachées, et une couche de sortie (Figure 13), donc la question qui se pose est comment réutilisé l'information précédente pour affecter l'information ultérieure ? La solution est d'ajouter une boucle qui retransmet l'information courante, donc la couche caché aura comme entrée une nouvelle donnée plus sa sortie elle-même (Figure 14).

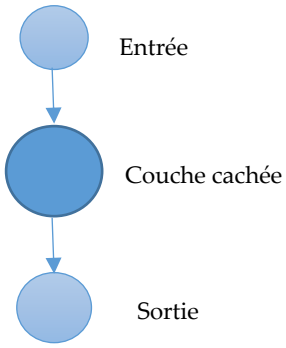


Figure 13 : Réseau neuronal *Feed Forward*

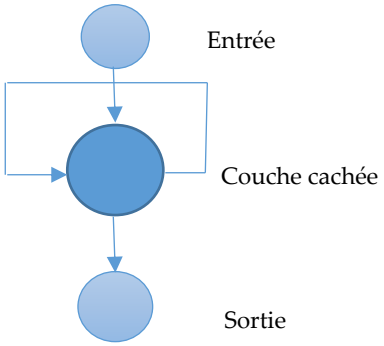


Figure 14 : architecture d'un RNN

Historiquement, ces réseaux ont été difficiles à entrainer, mais plus récemment, les progrès de la recherche au niveau de l'optimisation, les architectures de réseau, le parallélisme et unités de traitement graphique (GPU) ont rendus la construction de ces modèles possible.

Les deux premières architectures CNN et RNN, font partie de la catégorie des algorithmes supervisés, les deux architectures qui se suivent Aes et GANs, sont parmi les architectures les plus connues dans l'autre catégorie des algorithmes non supervisés.

3.3 Auto-encodeurs

Un AE est une approche de NN profond utilisée pour l'apprentissage non supervisé des caractéristiques avec un encodage et décodage efficaces des données. L'objectif principal d'un auto-encodeur est d'apprendre et de représenter (encoder) des données d'entrée, généralement pour la réduction de la dimensionnalité des données ou la compression, etc. Cette technique d'AE se compose de deux parties : L'encodeur et le décodeur, Dans l'encodage les échantillons d'entrée sont généralement 'traduits' dans un espace des caractéristiques dimensionnelles inférieures, cette approche peut être répétée jusqu'à ce que l'espace dimensionnel souhaité soit atteint. Dans la phase de décodage, nous régénérons les caractéristiques réelles à partir des caractéristiques dimensionnelles inférieures avec traitement inverse.

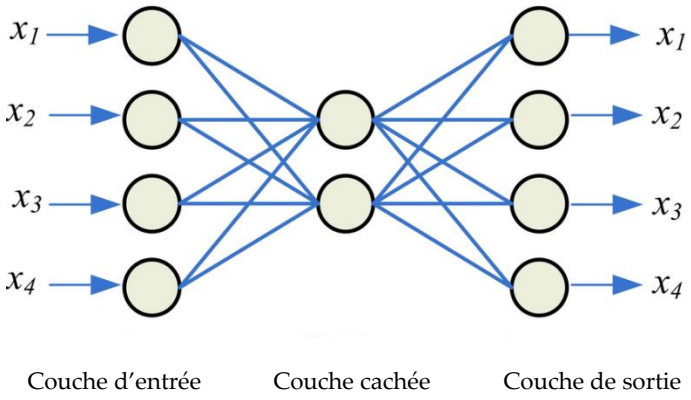


Figure 15 : l'architecture d'un AE [17]

Comme montré dans la Figure 15, nous pouvons prendre un ensemble de données non étiqueté et le traiter comme un problème d'apprentissage supervisé, en produisant une sortie très proche à celle d'entrée, c'est une reconstruction de l'entrée originale x. Ce réseau peut être formé en minimisant l'erreur de reconstruction, qui mesure les différences entre notre entrée originale et la reconstruction qui en résulte. On parlera plus sur les AEs dans [cf. Chapitre 3].

3.4 GANs

Une autre architecture qui a été considéré l'une des architectures les plus intéressantes dans le monde DL dans les dix dernières années est celle des GANs, c'est une approche

d'apprentissage profond récemment inventée par *Goodfellow* en 2014 [18] qui fait partie des algorithmes génératifs. Le GAN est un algorithme non supervisé où deux réseaux neuronaux s'affrontent, dans le cas du problème de génération d'images, le générateur commence par un bruit gaussien pour générer des images et le discriminateur détermine la qualité des images générées autrement dit. Il détermine si une image générée est très proche aux images réelles utilisées pour l'apprentissage, en se basent sur ce résultats, le générateur continue à s'améliorer jusqu'à ce que le discriminateur ne peut plus distinguer les images générées des images réelles. On a consacré toute une partie dans [cf. Chapitre 3] qui détaille cette architecture importante qui montre sa capacité à générer des images très proche à la réalité.

Conclusion

Dans la première partie de ce chapitre on a défini le transfert d'apprentissage, puis on a discuté la différence entre ce genre d'apprentissage et l'apprentissage automatique classique, et dans la partie de l'état de l'art, on a présenté quelques articles du transfert d'apprentissage et on a vu différentes applications dans différentes tâches.

Et comme on ne peut pas parler du transfert learning sans parler de DL, on a consacré la deuxième partie de ce chapitre pour en parler, voir ses différentes architectures, et se focaliser sur les CNNs, puisqu'ils sont les plus utilisés dans le domaine de la vision par ordinateur.

Les résultats obtenus par les différentes méthodes de transfert d'apprentissage et de l'adaptation de domaine sont atteints grâce à l'utilisation des réseaux de neurones profonds, le DL joue donc un grand rôle dans la construction de ces modèles de DA, particulièrement dans la phase d'extraction des caractéristiques. On parle donc de *Deep DA*.

Chapitre 3 : Notions dans la vision par ordinateur

Introduction

Ce chapitre parle des réseaux génératifs GANs, leur fonctionnement, et quelques cas d'utilisations dans la vision par ordinateur, ainsi, on explique l'approche du transfert d'apprentissage implémentée dans ce travail.

1 Image-to-image translation (ITIT)

ITIT est un sous-domaine qui appartient au monde de l'apprentissage automatique, ou plus précisément la vision par ordinateur *computer vision*. Il est devenu l'un des domaines les plus intéressants dans les dernières années, grâce à l'évolution des technologies de traitement, de stockage et d'analyse.

ITIT consiste à faire la conversion d'une image en entrée qui appartient à un domaine X, à une nouvelle image générée qui appartient à un autre domaine Y. En générale, le but est de construire des réseaux capables de faire le *mapping* entre les images d'un domaine source X et les images d'un domaine cible Y. Cela devient possible grâce à la naissance d'une nouvelle architecture de réseaux de neurones appelée *Generative adversarial networks (GANs)* [18]. La plupart des travaux implémentent cette architecture, qui a donné des résultats très satisfaisants.

2 Vision par ordinateur

La vision par ordinateur ou *Computer vision (CV)* est un domaine de recherche qui examine dans quelle mesure un ordinateur peut comprendre des images ou des vidéos numériques. Parmi les buts principaux de la vision par ordinateur est l'automatisation des tâches que les systèmes de vision humaine peuvent effectuer.

Les êtres humains utilisent leurs yeux et leurs cerveaux pour comprendre leurs entourages, donc on essaye de produire le même effet, à travers la vision artificielle pour que la machine puisse recevoir et comprendre l'image, pour qu'elle puisse réagir et prendre la décision. Les domaines d'application sont nombreux, la sécurité, la surveillance, le domaine médical, l'automobile, l'industrie, etc.

2.1 Applications de la CV

2.1.1 La détection d'objet

L'analyse d'une scène à partir des vidéos et des images numériques afin de détecter des objets dedans est parmi les tâches les plus compliquées dans le monde de la CV, ces objets peuvent être de différents types, tels que des êtres humains, des bâtiments, des voitures, etc. La détection d'objets a des applications dans de nombreux domaines de la vision artificielle,

elle est utilisée dans la détection des visages et la reconnaissance faciale. Elle est également utilisée pour suivre des objets, ou suivre une personne dans une vidéo.

Chaque classe d'objet a ses propres caractéristiques spéciales qui aident à la séparer des autres, donc pour détecter cette classe, il suffit de chercher ces caractéristiques, par exemple, lors de la recherche d'un cercle, on prend en considération que tous les cercles sont ronds.

Figure 16 : Résultats de détection de visage produits par Rowley. [19]

La détection de visage est l'une des applications les plus connues de la détection d'objet, c'est dû à son importance dans le processus de la reconnaissance faciale, comme nous le verrons dans le paragraphe suivant. La figure 16 montre le résultat du travail proposé dans [19], avec un seul faux positif parmi les 57 vrais positifs.

2.1.2 La reconnaissance faciale

La reconnaissance faciale est une méthode d'identification ou de vérification de l'identité d'un individu en utilisant son visage. Les systèmes de reconnaissance faciale peuvent être utilisés pour identifier des personnes sur des photos, des vidéos, récemment et avec l'amélioration des ressources, cette tâche peut être effectuée en temps réel.

La précision des systèmes de reconnaissance faciale s'est améliorée d'une façon remarquable, au point qu'ils deviennent préférés par rapport aux autres méthodes comme les empreintes digitales ou la reconnaissance de l'iris.

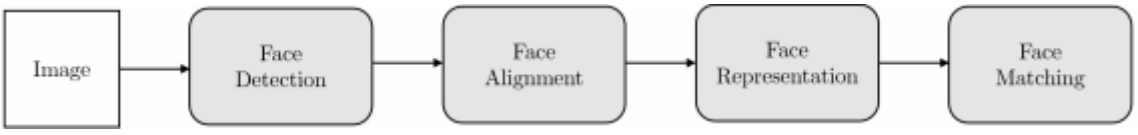


Figure 17 : les étapes principales dans un système de reconnaissance faciale [20].

La plupart des systèmes de reconnaissance faciale [20] passent par les étapes montrées dans la figure 17, on les détaille dans la partie suivante :

a) La détection faciale

Dans cette phase, l'algorithme essaye de localiser précisément la position du visage, cela se fait par la détection des yeux et d'autres points d'intérêts. La figure 18 montre un exemple d'image de sortie de cette phase.

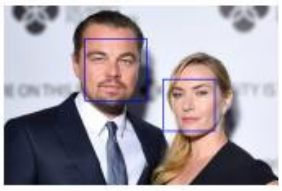


Figure 18 : Exemple de deux visages détectés et encadrés.

b) Alignement du visage

Cette étape consiste à récupérer la partie qui contient juste le visage détecté en effectuant un découpage, puis un redimensionnement, en se basant sur des points dans des positions fixées dans l'image. La figure 19 montre le résultat d'alignement des deux visages de la figure précédente, ainsi que les points références utilisés.

Figure 19 : deux images de visages alignés

c) Extraction des caractéristiques

Une transformation est effectuée au niveau des valeurs de pixels, pour aboutir à une représentation vectorielle de l'image. Ce vecteur représentatif doit être discriminant et contient que l'information pertinente. Normalement deux visages de la même personne doivent avoir deux vecteurs très proches.

d) Correspondance

La dernière phase consiste à calculer la distance entre deux vecteurs, et retourner un score, et en se basant sur ce score, le système donne une probabilité que les deux vecteurs appartiennent au même individu.

Les méthodes les plus récents de la reconnaissance faciale sont basées sur le DL. Ces méthodes ont montré leur efficacité et leur robustesse face aux variations tel que l'âge, l'expression faciale, la position du visage, etc.

2.1.3 Optical character recognition (OCR)

C'est un domaine très important qui commence à être déployé comme solution dans plusieurs applications, c'est le fait de reconnaitre le texte dans une image et l'extraire en format digital que la machine comprend, donc cette technologie donne la possibilité de convertir des documents de type diffèrent, (fichiers scannés, PDF, image, etc.) en texte que l'on peut manipuler. Atteindre une reconnaissance proche à celle de l'être humain reste toujours un grand défi, c'est pour cela les chercheurs dirigent leurs efforts vers ce champ de recherche pour améliorer leurs résultats. Dans [21], l'auteur mentionne que le problème d'OCR et complexe en raison de la diversité des langues, polices et styles dans lesquels le texte peut être écrit, et les règles complexes des langues, etc. Par conséquent, les techniques de différentes disciplines de l'intelligence artificielle sont employées pour traiter ces différents défis, traitement d'image, classification et traitement automatique de la langue naturelle (TALN), etc.

Nombreux sont les applications d'ORC, par exemples, la lecture de chèques bancaires et vérification de signature, traitement des formulaires disponibles sous forme imprimée, traitement des factures, validation des pièces d'identité, et reconnaissance automatique des plaques d'immatriculation, etc. [21]

Le mécanisme derrière un bon système d'OCRisation doit passer pas les étapes suivantes :

- **Le prétraitement de l'image** : l'image doit être traitée d'une manière avec laquelle on améliore sa qualité, les prétraitements les plus fréquents sont la réduction du bruit, l'alignement, la binarisation, etc.
- **La segmentation des caractères :** cette phase consiste à séparer les caractères dans l'image pour être adapter au système de reconnaissance, il existe plusieurs techniques de segmentation des caractères [22], [23].
- **L'extraction des caractéristiques :** Les caractères segmentés sont ensuite traités pour extraire différentes caractéristiques. Le modèle se base sur ces caractéristiques pour faire la reconnaissance.
- **La classification :** C'est la phase dans laquelle on doit attribuer une classe pour chaque caractère traité.
- **Le post-traitement :** Les techniques utilisées dans cette étape ont comme but principal l'amélioration du résultat, la précision n'est pas toujours 100%, donc on doit faire appel à plusieurs techniques comme TALN, ou d'autres programme qui sont basé sur des dictionnaires pour effectuer la vérification d'orthographe.

3 Generative adversarial networks (GANs)

3.1 Introduction

Comme noté dans l'introduction de ce chapitre, les GANs forment la base de ITIT, il s'agit d'une technique d'apprentissage automatique qui met en jeu une relation concurrentielle entre deux réseaux de neurones pour apprendre à générer de faux exemples qui « presque » ne se distinguent pas des données réelles, à partir d'un ensemble de données d'entraînement donné, par exemple, des images, des chiffres manuscrits, *etc.* L'architecture des GANs se compose de deux sous réseaux qui sont en « conflit » l'un contre l'autre, et c'est la raison pour laquelle on les appelle *adversarial* (contradictoire), cette architecture a été présenté la première fois par Ian Goodfellow *et al.* [18] en 2014, et elle a été considéré l'une des idées les plus intéressantes dans les dernières années. Ces types des réseaux ont la capacité d'imiter n'importe quelle distribution de données, ils sont donc capables de générer de nouvelle données à partir d'une base existante, dans n'importe quel domaine, la parole, l'image, la vidéo, *etc.*

Figure 20 : Images de haute résolution (1024 × 1024) générées à partir de la base de données CELEBA-HQ [23]

Les GANs n'étaient pas les premières architectures utilisées pour générer des données, mais leurs résultats les distinguaient des autres travaux, ces résultats remarquables ont été jugés pratiquement impossibles pour les systèmes artificiels, tels que la possibilité de générer de fausses images avec une qualité très proche à celle du monde réel. Figure 20.

Les deux Modèles qui forment les GANs sont :

- Modèle génératif : sert à générer des données qui ne se distinguent pas des données réelles.
- Modèle discriminatif : son but est de décider si un échantillon est réel, c-à-d appartient à la base d'apprentissage, ou cet échantillon est faux, et qu'il est généré par le Modèle génératif.

Pour mieux comprendre le fonctionnement des GANs, on doit d'abord parler brièvement des deux types de modèles qui forment leur noyau.

3.2 Modèles discriminatifs

Les algorithmes discriminatifs ont la tâche de classer les données d'entrée, c'est-à-dire que, à partir des caractéristiques d'une instance de données, ils prédisent une étiquette ou une catégorie à laquelle cette instance appartient. Dans [24], *Marc-Alexandre* mentionne qu'un modèle est dit discriminatif lorsqu'il représente la dépendance conditionnelle des variables non observées **y** (*labels*), par rapport à des variables observées **x** (*features*). Cela revient à modéliser la distribution de probabilités conditionnelles qu'on représente par $\mathbf{p(y|x)}$. Par exemple, étant donné tous les mots d'un courrier électronique (l'instance de données), un modèle discriminatif doit prédire si le message est un *spam* ou *non_spam*. *Spam* et *non_spam* sont des étiquettes ou les classes (**y**), et l'ensemble de mots dans un email sont les caractéristiques qui constituent les données d'entrée (**x**).

La nature prédictive des modèles discriminatifs permet de les privilégiés pour les tâches de classification et de régression. Parmi les algorithmes considérés discriminatifs, on retrouve la régression logistique, les forêts d'arbres décisionnels et les réseaux de neurones. [24]

3.3 Modèles génératifs

Le but d'un modèle génératif est en quelque sorte le contraire de celui du modèle discriminatif, il prend des données en entrée, par exemple quelques nombres aléatoires, et produit un résultat complexe, tel qu'une image d'un visage, ou une voiture, cela dépend des données d'entrainement.

Lorsque le modèle discriminatif représente les dépendances conditionnelles entre **y** et **x**, le modèle génératif représente les relations conjointes entre ces derniers, donc dans ce genre de réseaux, en cherche à modéliser la distribution de probabilités :

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x} \mid \mathbf{y})\, p(\mathbf{y}).$$

Lorsqu'on souhaite générer de nouvelles données, on fait appel aux modèles génératifs car ils ont la capacité de décrire la structure des données initiales, les données générer sont donc similaires à celles d'entrainement. Une façon de voir les algorithmes génératifs est que au lieu de prédire une étiquette en fonction de certaines caractéristiques, ils tentent de prédire des caractéristiques en fonction d'une certaine étiquette.

3.4 Fonctionnement des GANs

Comme mentionner dans l'introduction, les deux réseaux qui forment le *core* des GANs sont appelés *Generator* et *Discriminator.* Dans un premier temps, le générateur prend que du bruit en entrée, un vecteur des nombres aléatoires d'une dimension qui dépend du problème, (2D par exemple dans le cas des images), ensuite, il doit s'entrainer à générer des données d'aspect réaliste, il s'améliore en utilisant les résultats (*feedback*) retournés par le discriminateur. Par exemple, lorsqu'une image générer qui n'appartient pas aux données d'apprentissage est

classer par le discriminateur comme réelle (similaire à celles de la base d'apprentissage), le générateur sait qu'il a fait un bon travail, et chaque fois que le discriminateur rejette correctement une image produite par le générateur en tant que faux, le générateur sait qu'il doit être amélioré. Donc il doit prendre cela en considération lors de la mise à jour de ses poids.

Le discriminateur s'améliore également, pour chaque classification qu'il établit, il est informé si sa réponse est correcte ou non, au fur et à mesure que le générateur produit de plus en plus de données réalistes, le discriminateur s'entraine à mieux distinguer les fausses données du réelles.

3.4.1 Architecture

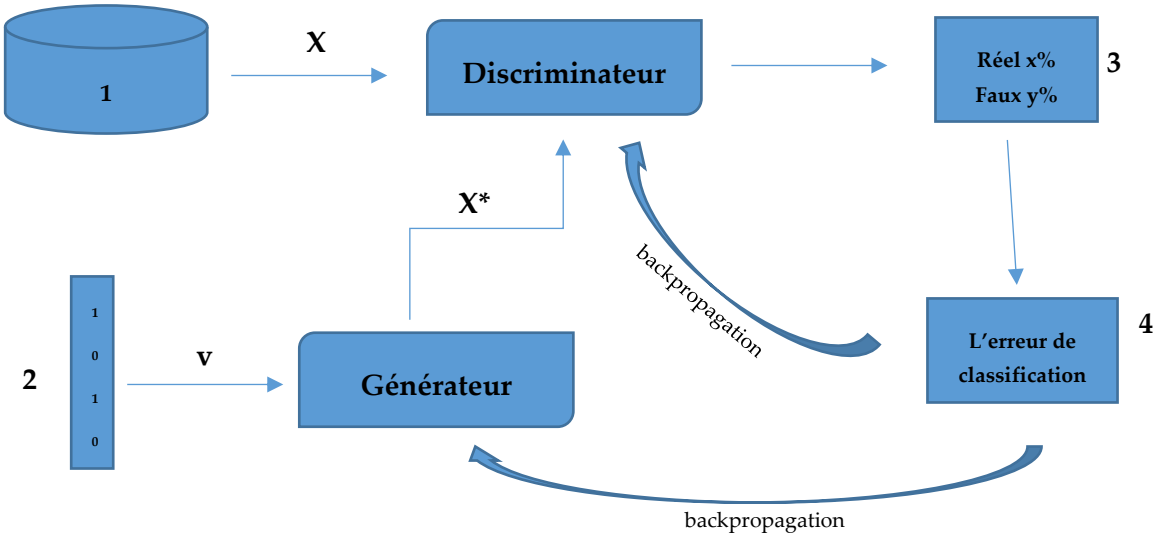


Figure 21 : Architecture générale d'un GAN.

- 1 : la base de données d'entrainement
- 2 : un vecteur de « bruit » utilisé par le générateur.
- Générateur : prend en entrée un vecteur aléatoire **v** pour générer un faux exemple.
- Discriminateur : prend en entrée un exemple réel **X** de la base de données d'apprentissage, et un faux exemple **X*** synthétisé par le générateur.
- 3 : la sortie du discriminateur, un pourcentage qu'un exemple appartient à la base d'apprentissage (réel) ou qu'il est généré (faux).
- 4 : on calcule l'erreur de classification (réel/faux), et avec la propagation arrière, on ajuste les poids des deux réseaux.

3.4.2 Algorithme

Pour** chaque itération **faire :

- *entrainer le discriminateur :*

a) *Sélectionner un exemple réel à partir de la base de données d'entrainement.*
b) *Générer un faux exemple à partir d'un vecteur aléatoire.*
c) *Utiliser le discriminateur pour classifier les deux exemples (**X** et **X***).*
d) *Calculer les erreurs de classification et faire une propagation arrière du total.*
e) *Mettre à jour les poids du discriminateur pour minimiser les erreurs de classification*

- *entrainer le générateur :*

- *Utiliser le générateur pour synthétiser un faux exemple à partir d'un vecteur de nombres aléatoires.*
- *Faire passer cet exemple au discriminateur pour la classification (réel ou faix).*
- *Calculer l'erreur de la classification et la propager en arrière.*
- *Mettre à jour les poids du générateur pour maximiser l'erreur du discriminateur.*

Fin Pour.

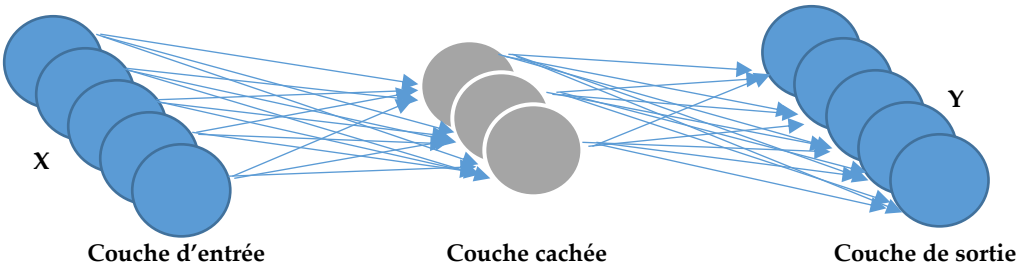
3.5 GANs & auto-encodeurs

Les GAN et les auto-encodeurs sont des modèles génératifs, mais ils ne sont pas utilisés de la même manière pour les mêmes taches. On va discuter la différence entre les deux modèles.

Un AE compresse son entrée en un vecteur, avec beaucoup moins de dimensions que la donnée d'entrée, puis le reconvertit en un vecteur de même forme que son entrée, en passant par plusieurs couches d'un réseau de neurones. Ils sont formés pour reproduire leurs entrées, c'est donc un peu comme un algorithme de compression ou de réduction de dimension. L'architecture des AEs est vraiment similaire à celle des réseaux de neurones feed-forward standard, mais l'objectif principal est différent. En raison de cette ressemblance, les mêmes techniques d'apprentissage peuvent être utilisées sur ce type de réseaux.

Les étapes d'apprentissage sont assez simples, les AEs peuvent être classés comme des algorithmes non supervisés (ils prennent les données d'apprentissage sans être étiquetées), mais qui utilisent des techniques d'algorithmes supervisés comme la propagation arrière. Un AE prend en entrée une instance de données, image par exemple, il encode cette instance sous forme d'un vecteur de dimension réduite, puis il essaye de reconstruire la même instance de

donnée d'entrée, il calcule la distance entre le résultat de reconstruction et l'instance réelle, et il effectue une propagation arrière de l'erreur pour le minimiser.



Dans l'architecture ci-dessus, un réseau simple qui dispose de 5 neurones en entrée, 3 neurones dans la couche cachée qui représente le vecteur des données compressé, et la couche de sortie qui a la même taille que celle d'entrée. $d(x, y) = ||x - y||$ est l'erreur qui représente la distance entre **x** et **y,** l'instance d'entrée et le résultat de reconstruction respectivement.

On a discuté dans [cf. Partie 3.4] que Les GANs adoptent une approche totalement différente. Ils utilisent un autre réseau (discriminateur) pour mesurer la distance entre les données générées et réelles. Il reçoit des données en entrée et renvoie un nombre compris entre 0 et 1. 0 signifie que les données sont fausses (générées) et 1 signifie qu'il s'agit de données réelles. L'objectif des générateurs est alors d'apprendre à convaincre le discriminateur de croire qu'il génère des données réelles.

En raison de ces différences, ils peuvent être utilisés pour différentes tâches. Les auto-encodeurs conviennent mieux à la compression de données dans des dimensions inférieures, où les GAN sont plus appropriés pour générer de nouvelles données.

3.6 Cas d'utilisation des GANs

Les champs d'application de l'*Image-to-image translation* deviennent de plus en plus large, en peux résoudre plusieurs problèmes en appliquant cette approche tel que la colorisation, le redimensionnement et plus d'autres exemples de la synthèse d'images.

3.6.1 Colorisation

La colorisation des images en niveaux de gris devient de plus en plus un champ de recherche très actif. Passer d'une image en couleurs à une image en niveaux de gris est une tâche très simple, il suffit d'extraire les trois composantes R, V, B de chaque pixel, ensuite calculer la valeur en niveau de gris, en utilisant une des plusieurs formules existantes, par exemple : gris = (R+V+B) /3.

Le problème se pose lors du passage inverse, puisqu'il existe plusieurs valeurs en couleurs pour une même valeur en niveau de gris. Il faut avoir une solution qui permet de choisir la couleur la plus adéquate pour remplacer un pixel en niveau de gris, cela exige la

compréhension du contenu de l'image. Un simple exemple est celui des visages, un algorithme de colorisation peut apprendre à déduire des couleurs dans des visages en niveaux de gris après avoir été entrainé à un jeu de données de visages en couleurs.

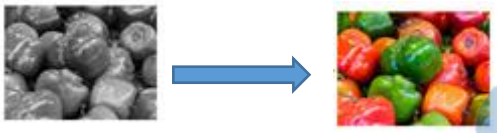


Figure 22 : colorisation d'une image en niveaux de gris.

3.6.2 Super-résolution

Un autre cas d'utilisation est celui qui permet d'obtenir une image de résolution supérieur à l'image réelle, l'image générée doit récupérer les détails perdus lors de la réduction de résolution. Ces détails sont alors déduits à partir d'une base d'apprentissage utilisée lors de la phase d'entrainement du réseau. Dans [25], les auteurs ont implémenté les GANs pour reconstruire une image en haute résolution à partir d'une image source de faible résolution, cela existait dans des travaux précédant, la valeur ajoutée par cette méthode abordée dans cet article est la récupération de textures les plus fines, ce qui donne des images de résolution x4 qu'on ne peut pas distinguer de l'image source donnée en entrée.

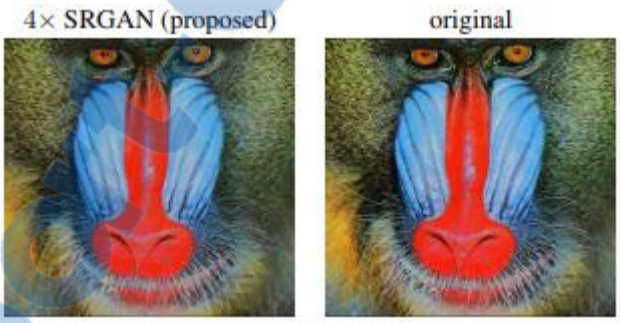


Figure 23 : à droite, l'image originale. A gauche, l'image générée par le réseau SRGAN proposé dans [25].

3.7 GANs pour l'augmentation des données

La masse des données joue un rôle très important lors de la phase d'entrainement d'un modèle, avoir une grande quantité de données aidera à construire un modèle robuste, mais cela n'est pas toujours le cas, parfois, nous devons faire face à des problèmes pour lesquels les données ne sont pas suffisante, l'une des méthodes les plus efficaces est l'augmentation des données (Data Augmentation). Le but est d'avoir une base d'apprentissage plus large à partir d'une base initiale moins convaincante.

Chercher plus d'échantillons n'est pas aussi facile que ça en a l'air, cela prend du temps, et parfois ces données ne sont pas même disponibles. Augmenter une base de données destinée

à l'apprentissage d'un modèle consiste à appliquer plusieurs opérations sur une seule image, le retournement, la rotation, le redimensionnement/mise à l'échelle, l'ajout du bruit, etc.

L'augmentation des données vient évidemment avant la phase d'apprentissage, mais il existe deux options principales :

La première consiste à réaliser toutes les transformations possibles avant de lancer l'apprentissage, c'est ce qu'on appelle **offline augmentation**. C'est une méthode destinée aux bases de données de taille limitée, pour la deuxième proposition, au lieu d'effectuer ces opérations sur toute la base de données, on augmente chaque échantillon à la volée juste avant de le faire passer au modèle, cette méthode est appelée **online augmentation** ou **augmentation on the fly** elle est destinée aux bases de données de grandes tailles.

3.7.1 Les techniques d'augmentation classiques

- Le retournement

Inverser les images est l'une des méthodes les plus populaires d'augmentation des données, Ceci est principalement dû à la simplicité de son implémentation. On peut retourner une image verticalement ou horizontalement.

Figure 24 : l'image originale suivie des deux retournements. [26]

- La rotation

C'est le mouvement circulaire de l'image autour du centre de rotation. Cette opération aidera le modèle à apprendre un objet dans n'importe quelle orientation.

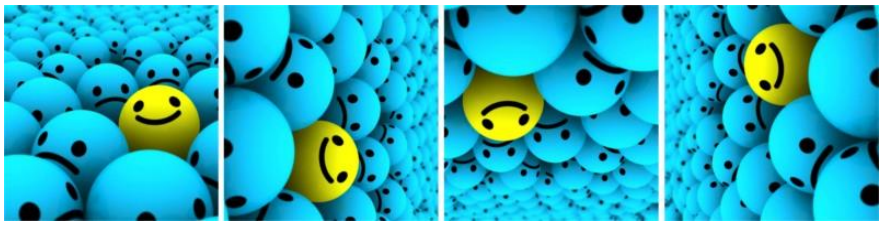


Figure 25 : l'image originale (à gauche) suivie de ses rotations. [26]

- Le redimensionnement

L'image peut être redimensionnée vers l'extérieur ou vers l'intérieur. Lors de la mise à l'échelle vers l'extérieur, la taille de l'image finale sera plus grande que la taille de l'image d'origine, ce problème peut être résolu par le découpage de la nouvelle image.

Figure 26 : l'image originale (à gauche) suivie de deux exemples de redimensionnement. [26]

- Le recadrage

Consiste à choisir aléatoirement une partie dans l'image originale, puis redimensionner cette zone à la taille initiale.

Figure 27 : l'image originale (à gauche) suivie de deux exemples de redimensionnement. [26]

- La translation

Déplacer l'image dans une direction donnée (x, y ou les deux), cette méthode est très importante, car les objets peuvent être placés n'importe où dans l'image, ce qui aide le modèle à chercher l'objet partout dans l'image.

Figure 28 : l'image originale (à gauche) suivie des translations dans les deux sens. [26]

Appliquer quelques changements aux données existantes permet d'élargir notre base d'apprentissage, cela conduit à construire un modèle robuste plus performant. Les méthodes qu'on vient de citer sont des méthodes classiques [26], alors que l'utilisation des GANs est une méthode plus avancée.

– **L'utilisation des GANs**

Dans de nombreux cas réels, nous devons atteindre des objectifs avec des ensembles de données limités, par exemple, dans le monde médical, les données sont fortement protégées en raisons liées à la protection de la vie privée. Ce qui fait que la collecte de données est une tâche fatigante, dans ce cas, les réseaux neuronaux profonds ne parviennent pas à donner le résultat désiré, certains des problèmes courants liés au fait d'avoir peu de données sont : le surapprentissage, et la production d'un modèle avec une mauvaise généralisation sur l'ensemble de test.

Nous venons de voir comment fonctionnent les GANs pour qu'ils puissent générer des données à partir des vecteurs de bruit aléatoires, en d'autres termes, ils ne font que mapper par exemple une matrice 28x28 de bruit aléatoire à une image très proche de ceux de notre réseau a été entrainé avec.

3.7.2 Application sur la base MNIST

Pour voir un exemple d'un réseau GAN en action, nous avons utilisé ce dernier pour générer plus d'échantillons de la fameuse base de données des chiffres manuscrites MNIST [27], la figure suivante montre les échantillons générés dans différentes itérations. L'apprentissage s'est fait en 30000 itérations, pendant environ 30 minutes avec une accélération GPU.

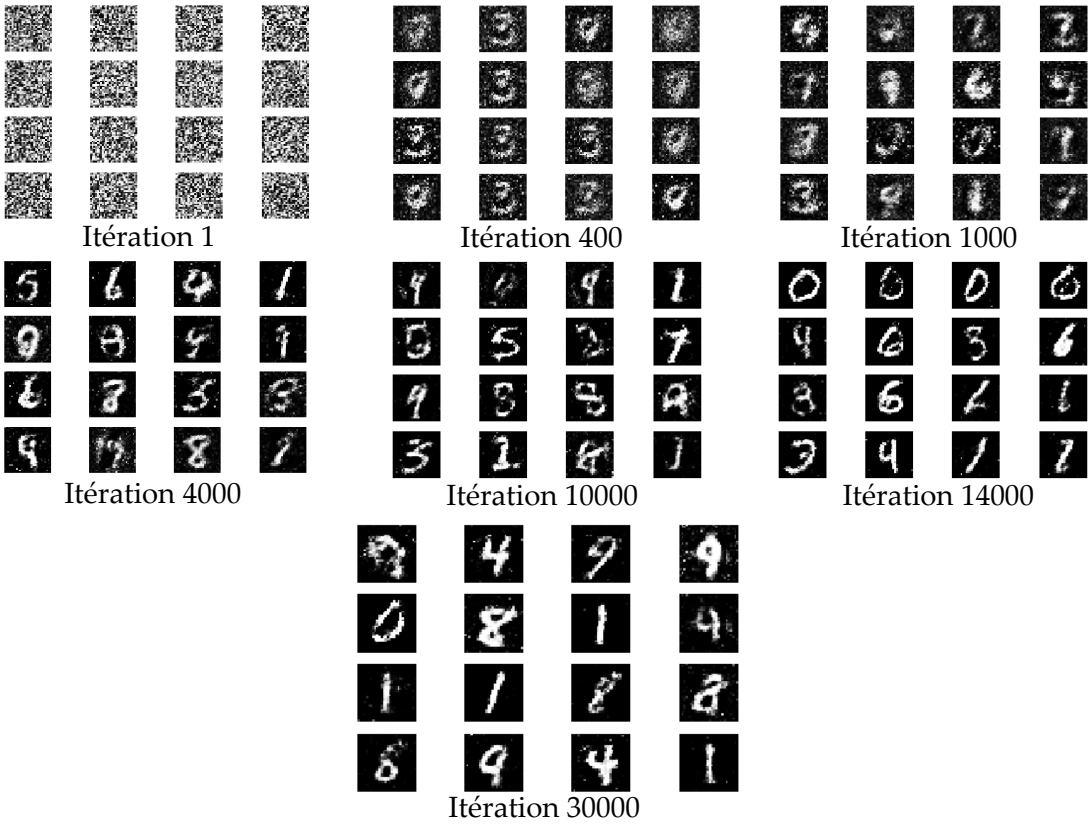


Figure 29 : Les échantillons générés par un réseau GAN.

Les GANs donc peuvent imiter n'importe quelle distribution de données, leur capacité de générer des images réalistes les a rendus le meilleur choix pour les chercheurs du monde de l'ITIT, afin qu'ils soient implémentés dans leurs applications.

4 Les approches de ITIT

Tout comme l'apprentissage automatique normal, ITIT peut également être divisé en deux catégories, ITIT supervisé et le non supervisé, dans ML, cela indique la présence ou bien l'absence des étiquettes pour les classes, alors que dans ITIT cela signifie si les deux bases de données source et cible sont liées ou pas [Cf. Chapitre 2].

La catégorie non supervisé reste dominante dans l'ITIT, c'est là ou se fait la plus grande partie de la recherche, l'une des principales raisons est la difficulté de collecter des données alignées dans les deux sources, autrement dit, pout chaque image dans le domaine source, on aura besoin de son image correspondante dans le domaine cible pour faire apprendre à un réseau de faire le mapping entre les deux.

Dans notre application, nous nous sommes concentrés sur des méthodes non supervisées, car les résultats préliminaires montrés par les méthodes supervisées n'ont pas été satisfaisantes. On détails le fonctionnement de quelques méthodes dans les parties suivantes de ce chapitre.

4.1 Unsupervised Image-to-Image Translation Networks (Unit)

Cette approche proposée par Ming-Yu *et al.* [28] fait partie de la catégorie non supervisée, les problèmes de cette catégorie sont considérés plus difficiles lors de l'entrainement des modèles, mais ils sont plus applicables puisque la collecte des données est plus facile. D'un point de vue probabiliste, le principal défi est d'avoir une distribution commune d'images dans différents domaines. Lorsqu'on parle de deux domaines différents, on parle de deux distributions marginales différentes, donc le but est de déduire une distribution conjointe pour les deux domaines, cela revient à faire le couplage des deux distributions, pour mieux comprendre cette notion de couplage. Supposons qu'on a deux variables aléatoires X et Y, le couplage de ces deux variables est un couple (X',Y') défini sur un espace commun, avec X' suit la loi de X, et Y' suit la loi de Y. une autre définition est que le couplage de deux variables aléatoires est une réalisation de ces variables sur un même espace et la dépendance entre les deux variables aléatoires permettra par exemple de les comparer [29]. Le problème qui se pose est qu'il existe une infinité de distributions communes qu'on peut atteindre à partir des distributions marginales données, déduire la bonne distribution veut dire avoir des d'hypothèses supplémentaires sur la structure de cette distribution conjointe.

Pour ce faire, les auteurs font une hypothèse d'un espace latent partagé, qui suppose qu'un pair d'images correspondantes dans différents domaines peut être mappé à une même

représentation latente dans cet espace partagé. Le modèle proposé nommé UNIT se constitue des GANs et des AEs, le rôle de l'AE est d'encoder les images des deux domaines dans l'espace partagé, les GANs vont ensuite générer les images dans les deux domaines à partie de cet encodage.

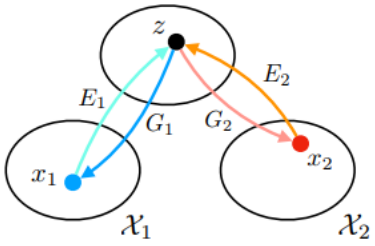


Figure 30 : l'espace caché z partagé où les images des deux domaines sont représentées. [28]

Les hypothèses de base des auteurs sont les suivantes ; X_1 et X_2 sont les deux domaines des images, pour x_1 et x_2 deux images appartenant à X_1 et X_2 respectivement, il existe une représentation latente partagée z dans un espace latent partagé (figure 30), de sorte qu'ils peuvent récupérer les deux images à partir de cette représentation. Cette représentation doit être atteinte à partie des deux images, les auteurs postulent qu'il existe des fonctions E_1, E_2, G_1 et G_2 tel que, données deux images x_1 et x_2, on a :

$z = E_1(x_1) = E_2(x_2)$ et

$x_1 = G_1(z)$ et $x_2 = G_2(z)$.

donc une fonction $F_{1\to2}$ définie par : $x_2 = F_{1\to2}(x_1)$ fait la traduction d'un domaine à un autre, elle peut être représenté par la composition des deux fonctions G_2 et E_1 :

$F_{1\to2}(x_1) = G_2(E_1(x_1))$

De la même façon on a :

$x_1 = F_{2\to1}(x_2) = G_1(E_2(x_2))$

Le but donc de cette approche est de faire l'apprentissage de $F_{1\to2}$ et $F_{1\to2}$. La récupération d'une image elle-même doit être possible en traduisant cette image d'entrée traduite, autrement dit :

$x_1 = F_{2\to1}(F_{1\to2}(x_1))$ et

$x_2 = F_{1\to2}(F_{2\to1}(x_2))$

Le modèle se compose de six sous réseaux, E_1 et E_2, les encodeurs des deux domaines, G_1 et G_2 les réseaux générateurs, et les deux discriminateurs D_1 et D_2. (Figure 31)

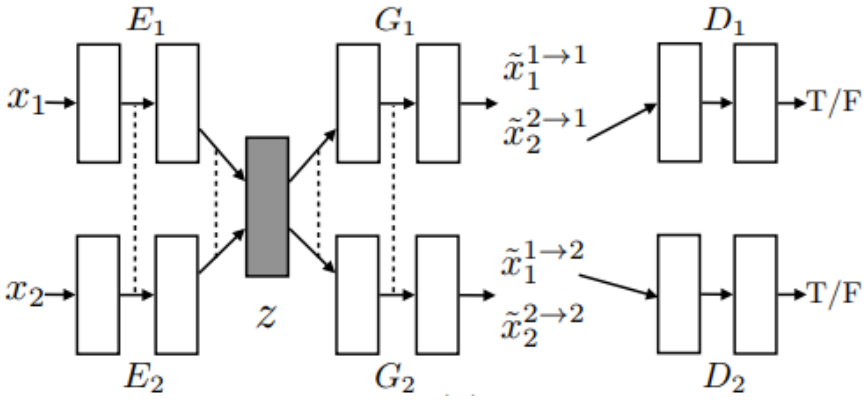


Figure 31 : l'architecture générale UNIT. [28]

Les auteurs représentent E_1 ,E_2 ,G_1 et G_2 en utilisant des CNNs, ils connectent les dernières couches de E_1 et E_2, et les premières couches de G_1 et G_2 , ce qu'ils appellent *weight-sharing* ou partage de poids. Ceci est fait pour relier les deux AEs, plus précisément, ils partagent les poids des dernières couches de E_1 et E_2 qui sont responsables de l'extraction des représentations de haut niveau des images d'entrée dans les deux domaines, et de la même façon, ils partagent les poids des premières couches de G_1 et G_2 chargées de décoder les représentations de haut niveau pour reconstruire les images d'entrée.

Les images générées $\tilde{x}^{1\to1}$ et $\tilde{x}^{2\to2}$ dont des images reconstruites du même domaine, alors que $\tilde{x}^{1\to2}$ et $\tilde{x}^{2\to1}$ sont des images traduites d'un domaine à un autre. D_1 et D_2 sont deux discriminateurs entrainés pour décider si les images générées sont réalistes ou pas.

4.1.1 Les AEs dans UNIT

Les deux réseaux E_1, G_1 forment le premier AE, noté AE_1, le rôle de AE_1 est premièrement d'encoder une image x_1 de X_1 et la représenter dans l'espace z à travers E_1, puis décode le code pour reconstruire l'image d'entrée via le générateur G_1 . De même, E_2, G_2 constituent un AE pour le domaine X_2 : AE_2 .

4.1.2 Les GANs dans UNIT

L'architecture se compose de deux réseaux GANs, GAN_1 contient D_1 et G_1. GAN_2 contient D_2 et G_2. Dans GAN_1, pour les images réelles échantillonnées à partir du premier domaine, D_1 doit donner vrai comme sortie, alors que pour les images générées par G_1, ce modèle doit retourner faux. G_1 doit générer deux types des images, celles qui viennent du même domaine $\tilde{x}^{1\to1}$, et celles traduites de l'autre domaine $\tilde{x}^{2\to1}$. l'apprentissage dans la partie de la reconstruction (1→1) peut être fait d'une manière supervisée, en calculant l'erreur à partir des images reconstruites et celles qui sont réelles. Donc l'apprentissage avec la méthode des GANs se fait juste dans la partie de la traduction 2→1 comme montré dans la figure 31.

4.1.3 Conclusion

Cette approche (UNIT) pose une hypothèse trop simplifiée, le modèle proposé effectue un mappage déterministe un à un, c'est l'un des limites de cette méthode, les auteurs ont mentionné qu'ils vont travailler là-dessus dans leurs futures recherches, et c'est exactement ce qu'ils ont fait dans leur prochain travail qu'on discutera dans la partie suivante.

4.2 Multimodal Unsupervised Image-to-Image Translation (MUNIT)

Une nouvelle approche présentée dans [30] par Ming-Yu *et al.* essaye de pallier aux insuffisances de UNIT [28], le mot *Multimodal* dans MUNIT exprime la capacité de ce modèle à générer plusieurs images en sortie à partir d'une seule image du domaine source, ce qui n'a pas été possible avec UNIT, ils supposent que la représentation de l'image peut être décomposée en un code de contenu et un code de style qui capture les propriétés spécifiques au domaine. Pour traduire une image d'un domaine à un autre, ils recombinent son code de contenu avec un code de style aléatoirement échantillonné de l'espace de style du domaine cible.

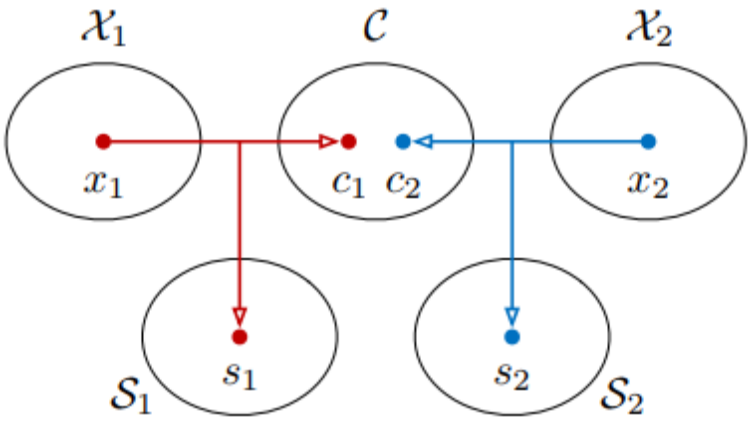


Figure 32 : l'hypothèse de l'espace caché ou sont représentées les images [30]

Contrairement à UNIT, les auteurs cette fois-ci suppose que l'espace caché où les images sont encodées peut être décomposé en deux sous espaces, l'un de contenu C, et l'autre de style S. Ils supposent en plus que les images dans différents domaines partagent l'espace de contenu commun mais pas l'espace de style. Traduire une image d'un domaine revient à combiner son code de contenu avec un code de style aléatoirement choisit de l'autre domaine cible. Le code de contenue contient l'information qui doit être conservée lors de la traduction, tandis que le code de style représente les variations restantes qui ne sont pas contenues dans l'image d'entrée. (Figure 32)

L'hypothèse cette fois-ci est d'avoir un espace latent partiellement partagé, plus précisément, les auteurs supposent que chaque image x_i appartenant au domaine X_i est générée à partir d'un code de contenu latent $c_i \in C$ qui est partagé par les deux domaines, et à

partir du code de style $s_i \in S_i$ qui est spécifique pour chaque domaine. Par exemple pour générer une image x_1 dans le domaine X_1, ils prennent son encodage de contenu c plus un code de style s_1 du domaine X_1 (figure 33). On a finalement :

$$x_1 = G_1(c, s_1) \text{ et } x_2 = G_2(c, s_2)$$

G_1 et G_2 sont deux générateurs dans les deux domaines X_1 et X_2 respectivement.

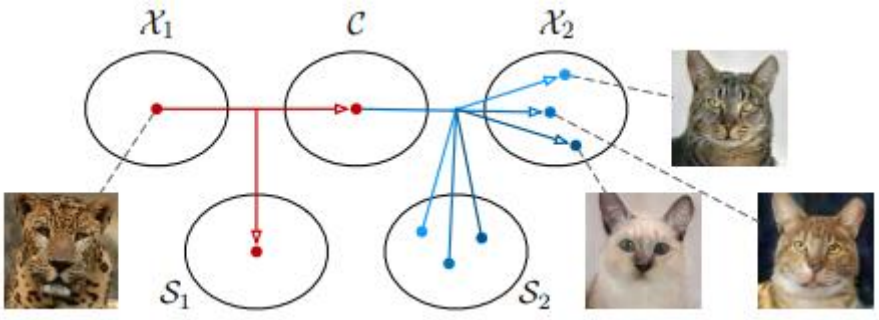


Figure 33 : La traduction d'une image (à gauche) du domaine X_1 en trois images du domaine X_2, par le choix aléatoire de trois styles différentes S_2 [30].

Les auteurs montrent la différence entre cette approche et celle de UNIT, ils disent qu'elle est étroitement liée à l'hypothèse d'espace caché partagé proposée dans UNIT, cette dernière suppose un espace entièrement partagé, alors que dans MUNIT, seule une partie de l'espace latent (le contenu) qui est partagée entre les domaines alors que l'autre partie (le style) est spécifique au domaine, ce qui est une hypothèse plus raisonnable lorsque le mappage inter-domaines est de plusieurs à plusieurs. [30]

Conclusion

Dans ce chapitre on a parlé de différentes notions et méthodes liées à notre sujet, on a consacré une grande partie pour parler des réseaux génératifs GANs, leur fonctionnement, et quelques cas d'utilisations dans la vision par ordinateur, cette partie aide à bien comprendre l'approche du transfert d'apprentissage implémentée dans ce travail.

Chapitre 4 : Expériences et résultats

Introduction

Dans ce chapitre, nous allons appliquer les méthodes précédentes sur notre cas, faire une comparaison des résultats obtenus par les différentes méthodes, nous allons présenter les résultats de la classification directe des images des montants manuscrits, et la comparer à celle des images des montants imprimés générées. On parlera aussi des outils et de l'enivrement de travail.

1 Environnement et préparation des données

Dans cette partie, nous verrons les éléments utilisés au cours de cette expérience, le logiciel et le matériel ou l'apprentissage du modèle a été effectuer, la base de données utilisée et comment elle a été générée.

1.1 Environnement de travail

1.1.1 Environnement logiciel

- **Langage de programmation**

Figure 34 : le logo python.

L'implémentation des modèles est faite par le langage Python, c'est le langage le plus adapté pour faire de l'apprentissage automatique, Créé par Guido van Rossum en 1991, Python est aujourd'hui un langage de programmation avancé et polyvalent. Sa syntaxe est plus lisible et simple, ce qui permet aux programmeurs d'écrire des modèles et des scripts en moins de lignes de code que d'autres langages comme C++ ou Java.

Cela fait de python un langage de programmation très populaire utilisé pour tous types d'applications : Développement Web, accès aux bases de données, IHM bureau, applications scientifiques et numériques, programmation réseau, développement de logiciels et de jeux, *etc*.

- **IDE utilisé**

Figure 35 : le logo Jupyter

Le Jupyter Notebook est une application web open-source qui permet de créer et de partager des documents contenant du code, des figures de visualisations et du texte, Le Notebook prend en charge plus de 40 langages de programmation, dont Python, R, *etc*.

Cette application client-serveur permet d'éditer et d'exécuter des documents de Notebook via un navigateur Web. L'application Jupyter Notebook peut être exécutée sur un ordinateur de bureau local ne nécessitant pas d'accès Internet ou peut être installée sur un serveur distant et être accessible par Internet.

- **Framework TensorFlow**

Figure 36 : logo TensorFlow

Créé par l'équipe Google Brain, TensorFlow est une bibliothèque open source pour le calcul numérique et l'apprentissage automatique à grande échelle. TensorFlow regroupe une multitude de modèles et d'algorithmes ML et DL et les rend facile à utiliser, Il utilise Python pour fournir une API front-end pratique pour construire des applications, tout en exécutant ces applications en C++ pour garantir une haute performance.

TensorFlow peut construire et exécuter des réseaux neuronaux profonds pour la classification et la reconnaissance d'images, des réseaux neuronaux récurrents, des modèles séquence à séquence pour la traduction automatique, le traitement du langage naturel, *etc*.

La structure principale des modèles construits par TensorFlow est basée sur des graphes c'est ce qu'on appelle des *dataflow graphs*. Les données se déplacent à travers ces graphes qui sont formés d'une série de nœuds de traitement. Chaque nœud du graphe représente une opération mathématique connue par le mot *Op*, et chaque connexion entre les nœuds est un

réseau de données multidimensionnel, ou tenseur, d'où le nom TensorFlow. Les opérations ne sont pas effectuées en Python, Les bibliothèques de transformations disponibles via TensorFlow sont écrites sous forme binaires C++, Python donc fournit des abstractions de programmation de haut niveau pour faciliter la tâche aux programmeurs. Les applications TensorFlow peuvent être exécutées sur la plupart des plateformes : une machine locale, une machine distante dans le cloud, des périphériques iOS et Android, et la plupart des fonctions ont deux implémentations, une pour être exécutées sur des CPU, et une autre pour les GPU, l'utilisation du GPU est l'une des raisons qui fait de TensorFlow le Framework le plus populaire dans le monde de ML, cela rend l'apprentissage machine pour les grand réseaux faisable dans une durée raisonnable.

- **TensorBoard :**

Figure 37 : logo TensorBoard.

Les calculs pour lesquels on utilise TensorFlow comme l'entraînement d'un réseau neuronal profond massif peuvent être complexes et déroutants. Pour faciliter la compréhension, le débogage et l'optimisation des programmes TensorFlow, l'équipe de TensorFlow ont inclus une suite d'outils de visualisation appelée TensorBoard. On peut utiliser TensorBoard pour visualiser les graphes TensorFlow, visualiser des courbes en temps réel lors de l'entrainement (l'évolution de l'erreur ou la précision par exemple) et afficher des données supplémentaires comme les images. On a utilisé TensorBoard pour avoir les courbes de l'erreur du modèle.

1.1.2 Environnement matériel

Pour entrainer des modèles qui traitent des images, on a besoin d'une capacité de calcules importante, cela nécessite l'utilisation GPU pour accélérer le calcule, et réduire le temps d'entrainement, l'utilisation d'une machine personnelle ne suffit plus, pour cela, on a travaillé sur un serveur propre à INDATACORE avec des capacités de calcule plus avancés, on accède au serveur par protocole SSH, et en utilisant l'application client-serveur Jupyter, le tableau suivant cite les caractéristique du serveur utilisé :

Machine	HP Z620 Desktop
RAM	64GB
Processeur	Intel(R) Xeon(R) CPU E5-2640 @ 2.50GHz * 24
GPU	GetForce GTX 1070/PCIe/SSE2
Stockage	2TB
OS	Ubuntu 16.04 LTS

Tableau 3 : caractéristiques du serveur utilisé pour l'implémentation.

1.2 Jeu de données

Comme déjà discuté dans les chapitres précédents, ce genre de modèle nécessite deux domaines pour apprendre à faire la traduction d'un domaine à un autre, on a besoin donc de deux bases de données, une pour les images des montants manuscrits (domaine cible), et une pour les images des montants imprimés (domaine source).

1.2.1 Base de données des montants manuscrits

Vu que c'est difficile d'avoir une base de données riche des montants manuscrits, on a décidé de générer notre propre base de données, à partir de plus de 30 exemples d'écritures des chiffes de 0 à 9 (figure 38).

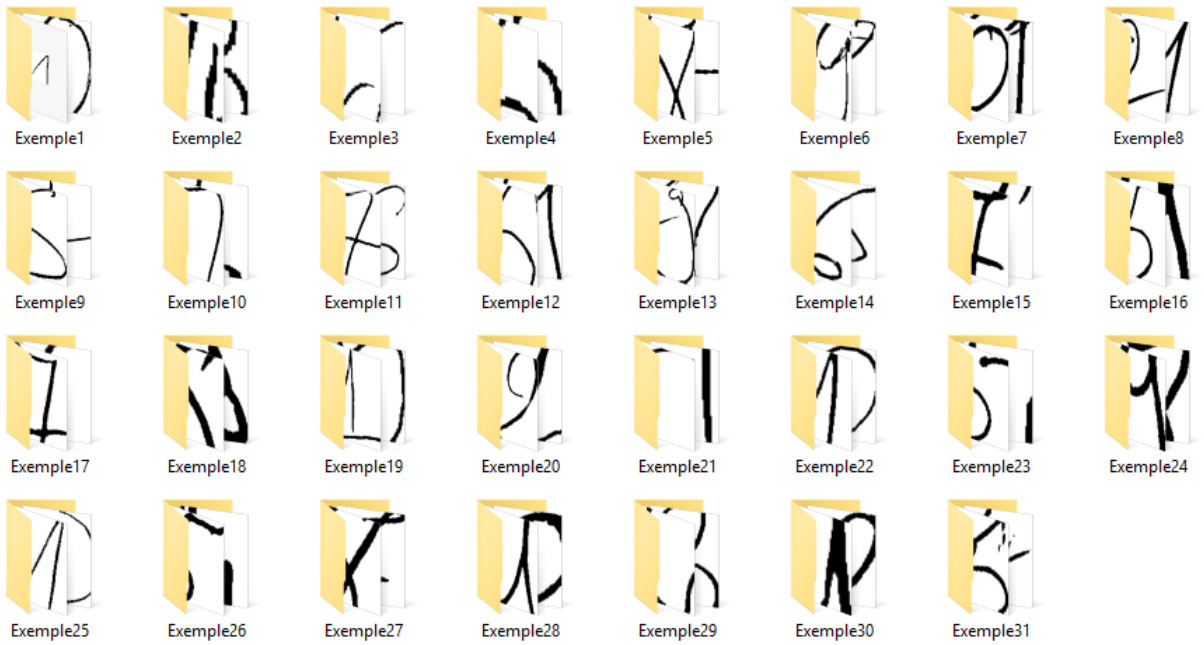


Figure 38 : exemples utilisés pour générer les montants manuscrits.

Chaque dossier contient une écriture différente des chiffres de 0 à 9, comme montré dans la figure suivante.

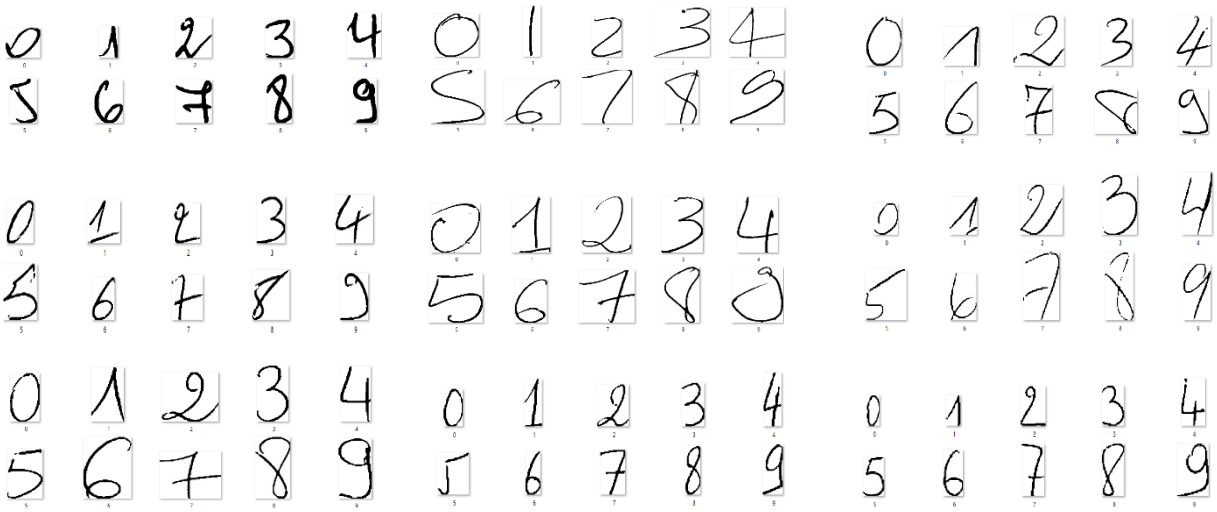


Figure 39 : échantillons des chiffres utilisés pour générer la base de données des montants manuscrits.

Après on a écrit un programme qui génère des séquences de 4 chiffres, en parcourant le répertoire de la figure 38, ce programme prend comme paramètres le chiffre de début (0 par exemple), le chiffre d'arrêt (9999 par exemple), et prend aussi le nombre d'échantillons par chiffre, (20 par exemple), pour ces paramètres on aura 20 différentes écritures pour chaque chiffre de 0 à 9999, donc un total de 200000 images seront générées.

Pour construire ces montants, on prend chaque chiffre, et on le met dans une image de taille 40x80, pour avoir un rembourrage qui est choisi aléatoirement pour différentes images, mais la taille finale de l'image doit être fixée, donc on aura les chiffres qui prennent différentes tailles dans un cadre de 40x80. La taille de la séquence générée est donc de 160x80 pour un montant de 4 chiffres, durant le redimensionnement des chiffres, on perd de l'information pour les chiffres minces (figure 40) donc, pour résoudre ce problème, on faisait appel à l'une des transformations morphologiques, on a effectué de l'érosion pour récupérer les pixels perdus lors du redimensionnement (figure 41).

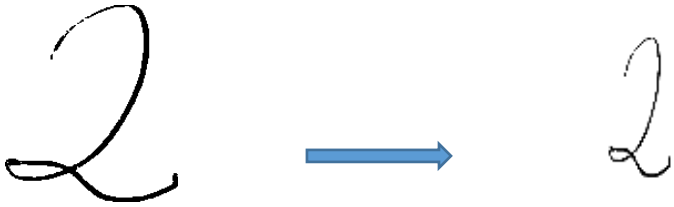


Image originale 156x177 Image après redimensionnement 40x80

Figure 40 : opération de redimensionnement avec perte de l'information.

Image avant érosion 40x80 Image après érosion 40x80

Figure 41 : opération de l'érosion.

Quelques exemples des images obtenues sont montrés dans le tableau suivant.

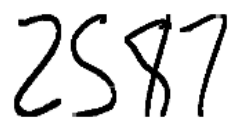
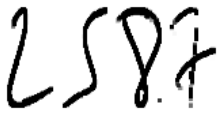
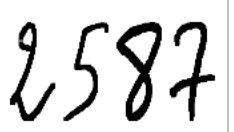
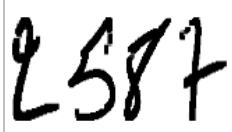
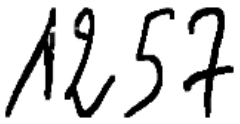
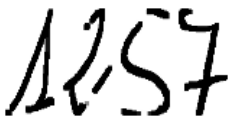
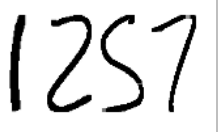
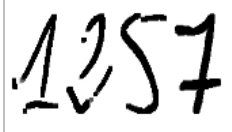
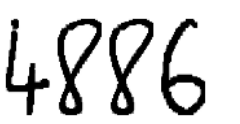
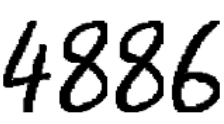
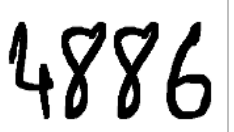
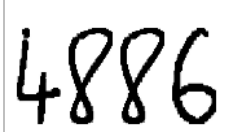
2587 :				
1257 :				
4886 :				

Tableau 4 : Exemples des images des montants manuscrits générées pour l'apprentissage.

1.2.2 Base de donnée des montants imprimés

Pour générer les ces images, on a utilisé une bibliothèque Python *text_to_image* qui permet d'avoir une image à partir du texte. On a écrit un script qui permet de générer des images des chiffres de 0 à 9 en utilisant plus de 90 différentes polices, c-à-d chaque chiffre doit être écrit en plusieurs polices, le répertoire de la figure 42 montre les sous-répertoires de chaque chiffre de 0 à 9 écrit 94 fois en différent polices.

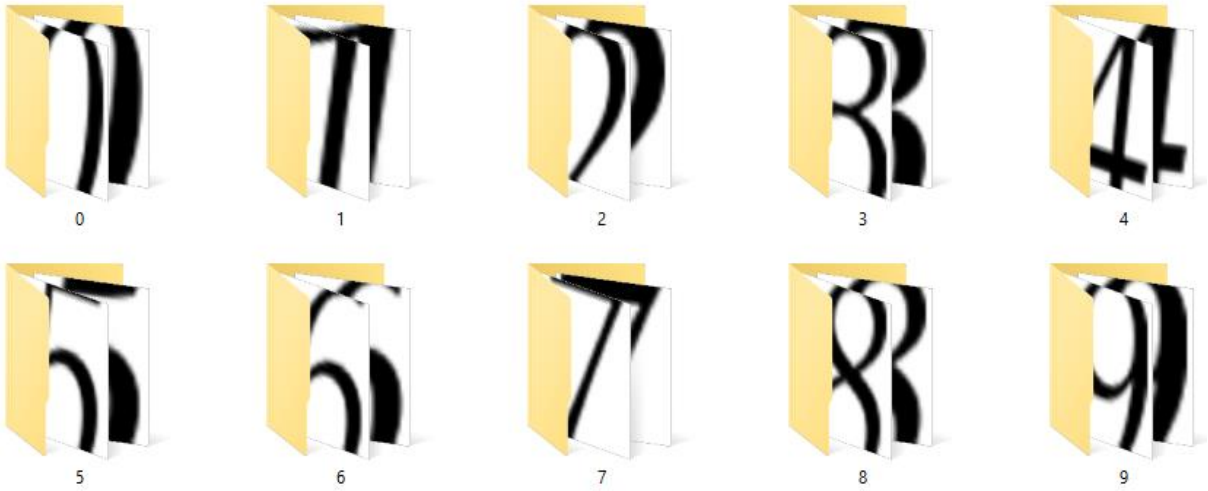


Figure 42 : Les répertoires des chiffres utilisés pour générer les images des montants imprimés.

La figure suivante montre quelques exemples de contenu des répertoires précédents.

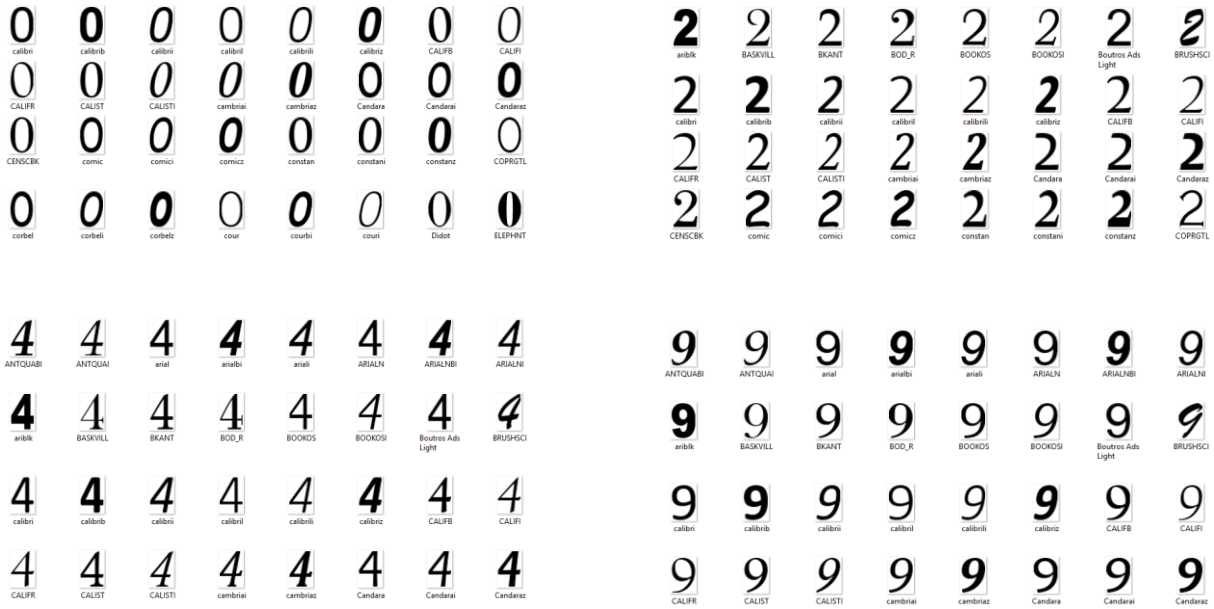


Figure 43 : Echantillons des chiffres utilisés pour générer la base de données des montants imprimés.

De la même manière, on construit des montants de 0 à 9999, 20 exemples pour chacun, ce qui donne 200000 images pour l'entrainement. Finalement on a un total de 4000000 images, 200000 pour chaque domaine.

2 Expérimentation

Le plan est d'entrainer un réseau de neurones constitué d'un CNN + RNN qui fait la reconnaissance des chiffres dans les images générer par le modèle MUNIT, donc le processus est le d'entrainer le modèle MUNIT puis passer à faire la reconnaissance avec le réseau LSTM. Le schéma suivant résume le processus proposé :

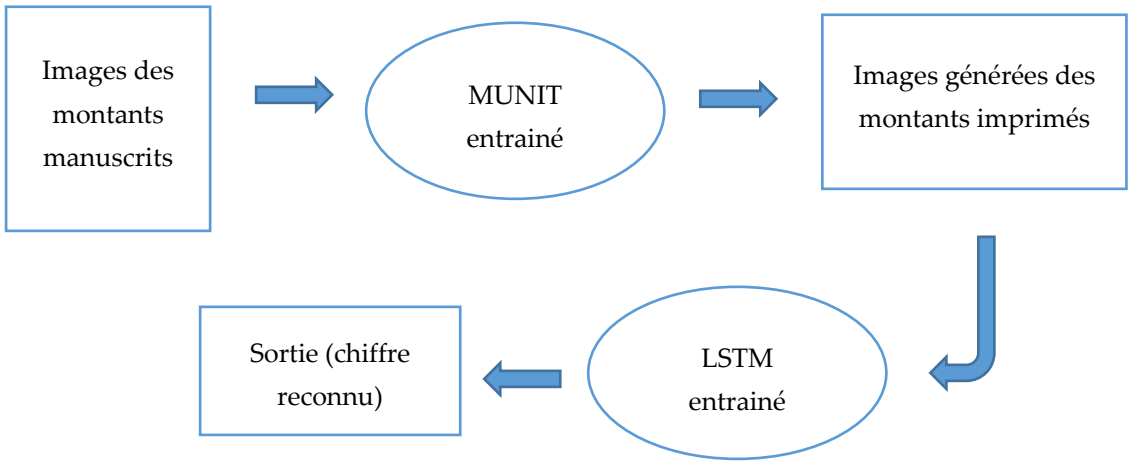


Figure 44 : Processus de la reconnaissance des montants.

2.1 Le modèle MUNIT

Durant la période des tests des différentes méthodes, on a décidé d'avancer avec la méthode MUNIT, cette méthode a été la plus convaincante au niveau de la qualité des images générées. Le modèle se compose de deux partie, la partie génératrice est composée d'un encodeur de contenu, un encodeur de style, et un décodeur pour les deux domaines, la partie discriminative de compose d'un réseau profond discriminatif [30].

2.1.1 Architecture du modèle

- Réseau génératif :
  - Encodeur de contenu :

Couche convolution de 64 filtres de taille 7x7

Couche convolution de 128 filtres de taille 4x4

Couche convolution de 256 filtres de taille 4x4

4 Couches résiduelles chacun contient 2 couches convolution de 256 filtres de taille 3x3

  - Encodeur de style :

Couche convolution de 64 filtres de taille 7x7

Couche convolution de 128 filtres de taille 4x4

3 Couches convolution de 256 filtres de taille 4x4

Couche de Pooling (*Average Pooling*)

Couche entièrement connectée

- Décodeur :

4 Couches résiduelles chacun contient 2 couches convolution de 256 filtres de taille 3x3

Couche redimensionnement x2 en utilisant *nearest-neighbor* approche

Couche convolution de 64 filtres de taille 5x5

Couche convolution de 3 filtres de taille 7x7

Image reconstruite

– Réseau discriminatif

Couche convolution de 64 filtres de taille 4x4

Couche convolution de 128 filtres de taille 4x4

Couche convolution de 256 filtres de taille 4x4

Couche convolution de 512 filtres de taille 4x4

2.2 Modèle LSTM

Ce réseau est constitué d'un CNN, qui fait l'extraction des caractéristiques de l'image, puis un RNN (LSTM) et une couche de sortie CTC (*Connectionist Temporal Classification*) qui prend en charge la sortie du RNN pour calculer l'erreur, ou décoder le montant contenu dans l'image.

2.2.1 Architecture LSTM

Le modèle LSTM se compose de 5 couches CNN, 2 couches RNN (LSTM) et de la couche de décodage CTC.

2 couches convolution de filtre de taille 5x5

3 couches convolution de filtre de taille 3x3

Couche activation RELU

Couche de Pooling (*max Pooling*)

2 couches LSTM avec 256 unités

Couche CTC (calcule l'erreur lors de l'entrainement, décode le montant lors du test)

3 Résultats

Commençant par les résultats obtenus de l'entrainement du modèle MUNIT visualisé par le TensorBoard, les courbes suivantes montrent l'évolution de l'erreur de la discrimination entre les images réelles, et celles générées par le modèle pour les deux domaines avec une courbe de la somme des deux erreurs. Cette erreur montre à quel point les images générées par le model peuvent être comparables à celles qui sont réelles, car cette erreur calcule la différence entre chaque image générée et l'image réelle correspondante, prenons par exemple deux images x_1 et x_2 correspondantes des deux domaines X_1 et X_2, l'erreur montrée dans les courbes est de x_1 et $x_{2\to1}$, avec $x_{2\to1}$ l'image traduite du domaine X_2 vers X_1.

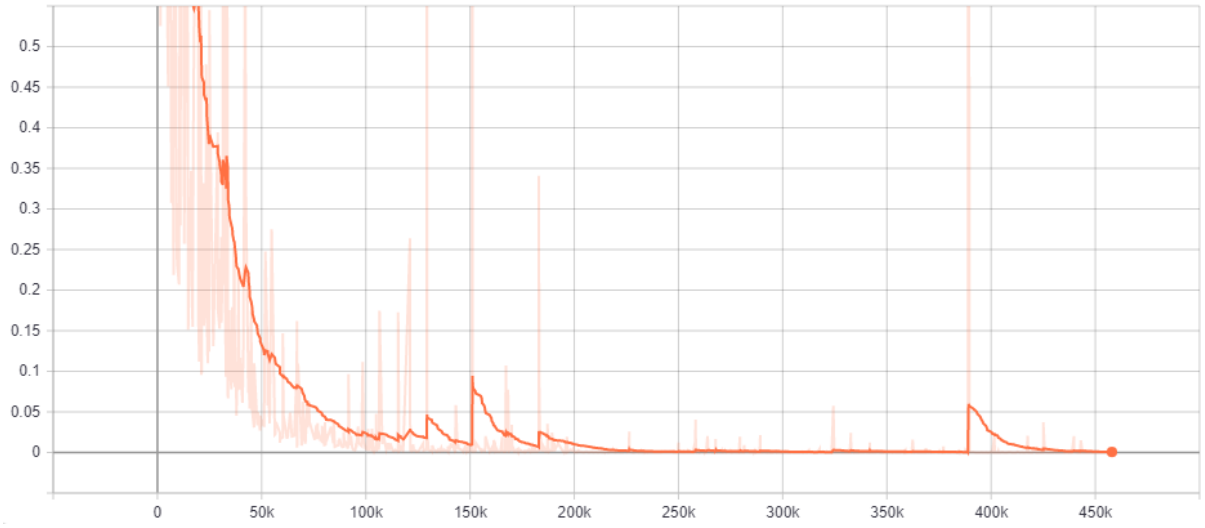


Figure 45 : la courbe de l'erreur pour les images manuscrites générées.

Dans la figure 45, on remarque que l'erreur calculée entre les images générées et les images réelles dans le domaine manuscrit tend vers 0. Cela veut dire que le modèle génère des images des montants manuscrits d'une façon presque parfaite.

La courbe suivante montre l'évolution de l'erreur lors de l'entrainement du modèle dans le cas des images des montants imprimés.

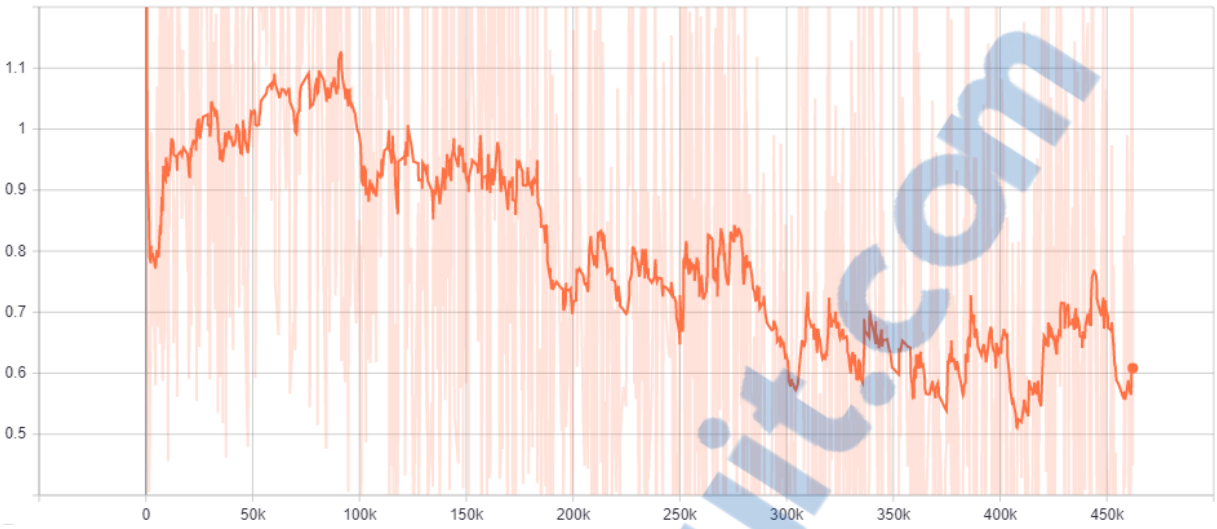


Figure 46 : la courbe de l'erreur du domaine source (montants imprimés).

Pour ce domaine, l'erreur bascule à peu près entre 0.5 et 0.7, ce qui montre que les images générer sont comparables à celles qui sont réelles. La courbe suivante montre l'évolution de l'erreur générale.

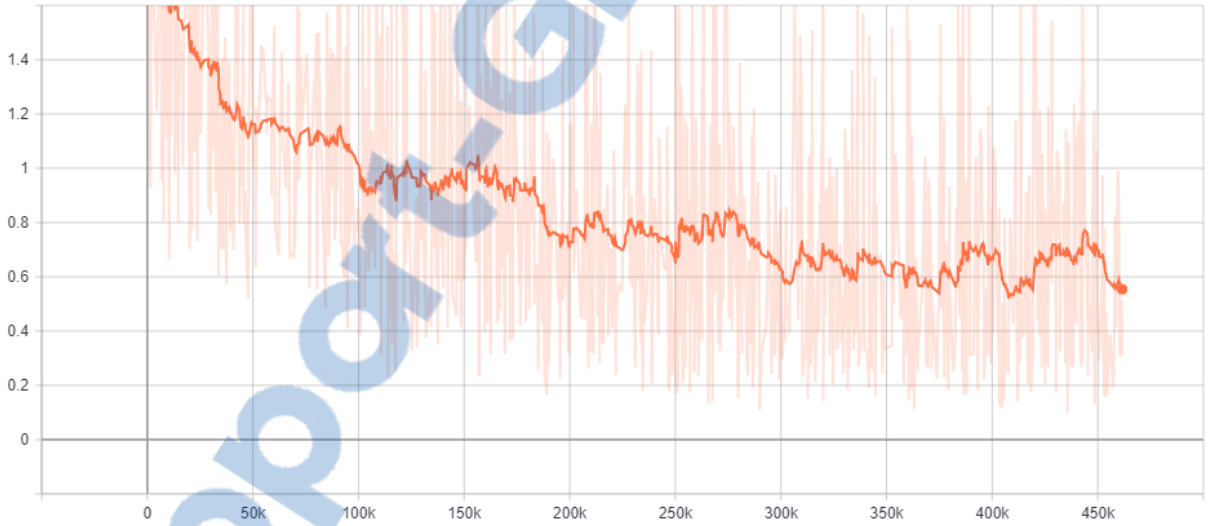


Figure 47 : la courbe de l'erreur totale des deux domaines.

3.1 Visualisation des résultats

Maintenant on passe à la visualisation des résultats de la traduction des images des montants manuscrits vers les images des montants imprimés. Le tableau suivant montre d'abord les résultats de la traduction dans les premières itérations d'entrainement :

	Image à traduire (montant manuscrit)	Image générée (montant imprimé)
Itération 1000 :		
Itération 2000 :		
Itération 6000 :		

Tableau 5 : résultats de traduction dans les premières itérations.

On remarque bien que le modèle commence à apprendre comment générer des images des montants imprimés à partir de ceux manuscrits. Le plus qu'on avance dans les itérations, le plus les images générées tendent à être de bonne qualité.

Le tableau suivant montre quelques exemples de la traduction parfaite des montants manuscrits en montant imprimés lors de la phase de test. Ce sont les images qu'on doit utiliser dans l'étape suivante de la reconnaissance avec le réseau LSTM.

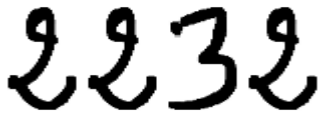
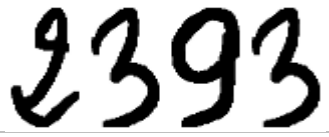
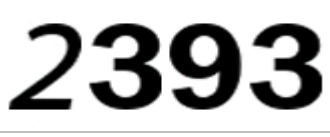
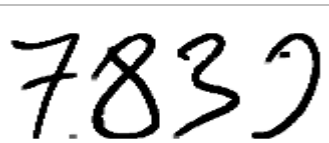
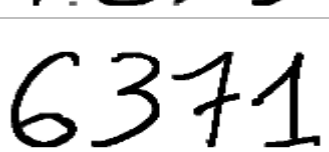
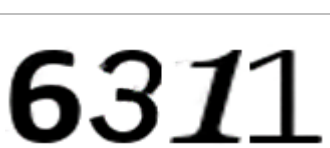
Image à traduire (montant manuscrit)	Image générée (montant imprimé)	Précision par chiffre
		**4/4**
		**4/4**
		**4/4**
		**4/4**
		**3/4**
		**3/4**
		**3/4**
		**3/4**
		**3/4**

Tableau 6 : visualisation des images générées lors du test.

A partir de ces résultats on remarque que le modèle arrive à imiter les montants imprimés à partir des montants manuscrits, les images générées peuvent être facilement reconnues par n'importe quel modèle d'OCR pré-entrainé.

3.2 Résultats LSTM

On a entrainé le réseau LSTM en utilisant les images des deux bases de données (deux domaines), ensuite on a testé ce réseau en utilisant les images générées par le modèle MUNIT, et pour savoir si cette méthode donne de meilleurs résultats, on a comparé ses résultats avec les résultats donnés par l'utilisation des montants manuscrits directement sans faire la traduction vers le domaine imprimé. Le tableau suivant montre le pourcentage des montants reconnus pour les deux bases de test, ainsi que le pourcentage des chiffres non reconnus.

	Pourcentage des montants reconnus (précision)	**Pourcentage des chiffres non reconnus (taux d'erreur)**
Utilisation directe des images manuscrits	80.461538 %	7.444169 %
Utilisation des images traduites (méthode proposée)	92.650000 %	2.681661 %

Tableau 7 : résultat de la comparaison des deux approches.

D'après ces résultats, on remarque que l'utilisation des images traduites a surpassé l'utilisation directe des images extraites du chèque, le taux de reconnaissance de la première atteint 92%, avec 12% de mieux que la deuxième (80%).

Conclusion

Dans ce chapitre, on a vu les différentes étapes de la préparation des données, ainsi que l'environnement on se déroulé l'apprentissage avec les outils utilisés, puis on a visualisé les images générées par le modèle implémenté, on a présenté aussi les résultats de la comparaison entre la méthode proposée et celle existante, on a démontré que notre méthode donne une meilleure précision avec un taux d'erreur moins élevé.

Conclusion générale

Ce travail a été réalisé dans le but de faire face au problème de la reconnaissance des chiffres manuscrits à partir des images des chèques, les caractères écrits à la main font passer les choses à un autre niveau de complexité. Contrairement aux polices standards, le texte manuscrit contient rarement des modèles réguliers ou prévisibles, ce qui rend la tâche plus complexe.

Pour cela, on a décidé de chercher une manière pour adapter une méthode pour traduire les images des montants manuscrites en des images des montent imprimées, cela nous a poussé à creuser dans le monde de transfert d'apprentissage ou *transfer learning* (TL), et faire une étude sur les différentes méthodes existantes qui permettent de traduire les images des chiffres manuscrites en images des chiffres imprimées. Donc au lieu de faire la reconnaissance directe du chiffre manuscrits, on le fait passer dans un modèle qui généré son correspondant imprimé, cela revient à faire de l'OCR du texte imprimé. On a montré dans la synthèse que cette approche a donnée de meilleures résultats que celle qui utilise la reconnaissance directe des montants manuscrits.

Le problème avec les réseaux génératifs est qu'ils nécessitent un temps important dans la phase d'apprentissage, et des ressources matérielles puissantes qui accélèrent l'apprentissage, donc l'utilisation d'une machine bien équipée est très important pour rendre l'apprentissage possible. Un autre problème qu'on a rencontré est le manque d'une base de données réelle des montants, ce qui nous a poussé à générer une base avec les exemples qu'on a pu collecter.

Ce travail va être poursuivis afin d'améliorer le modèle génératif, qui trouve parfois des difficultés à distinguer quelques chiffres, on essayera de régénérer plusieurs bases de données avec différentes polices, aussi on essayera de focaliser l'entrainement sur ces chiffres mal reconnus.

Bibliographie

[2] S. Pan et Q. Yang, "A Survey on Transfer Learning", *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, 2010.

[3] Wouter M. Kouw, Marco Loog, Yang, " An introduction to domain adaptation and transfer learning", arXiv:1812.11806, Dec. 2018. [En ligne]. Disponible : https://arxiv.org/abs/1812.11806.

[4] Eric Tzeng *et al,* " Adversarial Discriminative Domain Adaptation", arXiv:1702.05464 [cs.CV], Feb. 2017. [En ligne]. Disponible : https://arxiv.org/abs/1702.05464.

[5] Tzeng, *et al*, " Simultaneous Deep Transfer Across Domains and Tasks". *2015 IEEE International Conference on Computer Vision (ICCV).*2015.

[6] Meina Kan *et al*, " Bi-shifting Auto-Encoder for Unsupervised Domain Adaptation", *2015 IEEE International Conference on Computer Vision (ICCV).*Feb 2016.

[7] Sungeun Hong *et al*," SSPP-DAN: Deep Domain Adaptation Network for Face Recognition with Single Sample Per Person", arXiv:1702.04069 [cs.CV] Apr 2018. [En ligne]. Disponible : https://arxiv.org/abs/1702.04069.

[8] Philip Isola *et al*," Image-to-Image Translation with Conditional Adversarial Networks", arXiv:1611.07004, Nov2018. [En ligne]. Disponible : https://arxiv.org/abs/1611.07004.

[9] Sungeun Hong *et al*," Adapting Deep Visuomotor Representations with Weak Pairwise Constraints", arXiv:1511.07111, May 2017. [En ligne]. Disponible : https://arxiv.org/abs/1511.07111.

[10] Murez, Z., Kolouri, S., Kriegman, D., Ramamoorthi, R., & Kim, K. (2018). " Image to Image Translation for Domain Adaptation ". *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 2018.

[11] Adam Gibson, Josh Patterson, "A Review of Machine Learning" Dans Deep learning, édit. O'Reilly Media, Inc. 2017.

[12] Fukushima, K. Neocognitron. "A hierarchical neural network capable of visual pattern recognition". *Neural Netw*. 1988, 1,P. 119–130

[18] Ian J. Goodfellow *et al*," Generative Adversarial Networks", arXiv:1406.2661 [stat.ML], Jun 2014. [En ligne]. Disponible : https://arxiv.org/abs/1406.2661.

[19] Rowley, H., Baluja, S., et Kanade, T. "Neural network-based face detection". *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*1996.

[20] Daniel Sáez Trigueros *et al*," Face Recognition: From Traditional to Deep Learning Methods", arXiv:1811.00116, Oct 2018. [En ligne]. Disponible : https://arxiv.org/abs/1811.00116

[21] Noman Islam *et al* "A Survey on Optical Character Recognition System", *Journal of Information & Communication Technology-JICT* Vol. 10 Issue. 2, Dec 2016.

[22] Mohammed Javed *et al*," Extraction of Line Word Character Segments Directly from Run Length Compressed Printed Text Documents", arXiv:1403.7783 [cs.CV], Mar 2014. [En ligne]. Disponible : https://arxiv.org/abs/1403.7783

[23] Mohammed Javed *et al*," Progressive Growing of GANs for Improved Quality, Stability, and Variation", arXiv:1710.10196 [cs.NE], Feb 2018. [En ligne]. Disponible : https://arxiv.org/abs/1710.10196

[24] Marc-Alexandre Côté " RÉSEAUX DE NEURONES GÉNÉRATIFS AVEC STRUCTURE". *Thèse présentée au Département d'informatique en vue de l'obtention du grade de philsophiæ doctor (Ph.D.)* Avr 2017.

[25] Christian Ledig *et al*," Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network", arXiv:1609.04802 [cs.CV], May 2017. [En ligne]. Disponible : https://arxiv.org/abs/1609.04802

[28] Ming-Yu Liu *et al*," Unsupervised Image-to-Image Translation Networks", arXiv:1703.00848 [cs.CV], Jul 2018. [En ligne]. Disponible : https://arxiv.org/abs/1703.00848

[29] Francis Comets, " Couplage et Chaînes de Markov" Dans chaînes de Markov, Nov 2017.

[30] Xun Huang *et al*, "Multimodal unsupervised Image-to-Image Translation ", arXiv:1804.04732 [cs.CV], Aug 2018. [En ligne]. Disponible : https://arxiv.org/abs/1804.04732

Webographie

[1] https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a : Introduction vers le transfert d'apprentissage avec quelques applications.

[13] https://skymind.ai/wiki/convolutional-network : Guide pour une compréhension simple des CNNs.

[14] http://cs231n.github.io/convolutional-networks/ : Les couches CNN expliquées en détails.

[16] https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148 : Article web sur les CNNs.

[17] https://www.researchgate.net/figure/The-structure-of-a-four-input-four-output-autoencoder_fig2_265497360 : Architecture basique des Auto-encodeurs.

[26] https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced : Augmentation des données.

[27] http://yann.lecun.com/exdb/mnist/ : Site officiel de la base MNIST.

[31] https://www.python.org/ Site officiel du langage Python.

Extraction des connaissances à base des approches "Transfer Learning" et application dans la vision par ordinateur.

Résumé

L'extraction des connaissances est la création des connaissances à partir des sources structurées comme des bases de données, ou des sources non structurées comme des images. L'extraction de l'information à partir des images est un nouveau défi dans le domaine de la vision par ordinateur, l'une des utilisations importantes de l'extraction des connaissances est l'automatisation de plusieurs tâches comme la lecture des chèques bancaires et l'extraction automatique des informations d'identification dans les documents personnels comme la CIN, Passeport, etc. Pour le texte imprimé, il existe des modèles pré-entrainés capables d'extraire le texte à partir des images, contrairement au texte manuscrit, qui reste difficile à reconnaitre, ce qui en fait un grand champ de recherche. Ce travail fait partie de ce champ de recherche, dans lequel on propose une méthode d'extraction des montants manuscrits à partir des chèques bancaires, en montrant que la méthode proposée donne de meilleurs résultat.

Mots clés : Extraction des connaissances, OCR, Vision par ordinateur.

Knowledge extraction using transfer learning with application in computer vision

Abstract

Knowledge extraction is the creation of knowledge from structured sources such as databases, or unstructured sources such as images. Extracting information from images is a new challenge in the field of computer vision, one of the important uses of knowledge extraction is the automation of several tasks such as reading bank cheques and automatically extracting identification information from personal documents such as ID cards, Passports, etc. For printed text, there are pre-trained models that can extract text from images, unlike handwritten text, which is still difficult to recognize, making it a large field of research. This work is part of this field of research, in which a method is proposed for extracting handwritten amounts from bank cheques, showing that the proposed method gives better results.

Keywords : knowledge extraction, ORC, Computer vision.

Master Systèmes Intelligents & Réseaux

Département d'Informatique

Faculté des Sciences et Techniques de Fès

A.U. 2018 - 2019