Projet 7 : Developpez une preuve de concept

Maby Antoine
Décembre 2022

Sommaire

1 Introduction **2**

2 Etat de l'art du self surpervised learning **3**

3 Choix des données et Baseline **5**

4 Description de SimCLR **6**

5 Modélisation **8**
5.1 Inceptionv3 8
5.2 SimCLR avec pré-entrainement sur les photos de chiens 8
5.3 SimCLR avec pre entrainement sur ImageNet 9

6 Conclusion **10**

Chapitre 1

Introduction

Le domaine du machine learning, et plus généralement de la data science, évolue très rapidement. Il est alors important de se tenir au courant des avancées dans le domaine en effectuant une veille thématique. Nous devons aussi être capable de monter rapidement en compétence sur une nouvelle thématique, en sachant effectuer une recherche et mettre en pratique un nouvel algorithme de façon autonome (POC). Le sujet de projet est donc un entrainement à cette problèmatique.

Ansi, dans le but de réaliser ce projet, j'ai décidé de me focaliser sur l'apprentissage non supervisé de la classification des images. Dans le projet précédent, nous avons réalisé un réseau de neuronnes convolutionnels pour déterminer les races de chiens. Pour cela, nous avions a disposition une database de plus de 20 000 images déjà classifiées pour l'apprentissage puis pour le test. Malheureusement, dans la réalité, nous n'aurions pas forcement eu à disposition une si grande base de données avec toutes les images déjà étiquetées. C'est sur cette problématique que j'ai décidé de me focaliser.

L'apprentissage auto surpervisé est justement une solution pour résoudre notre problème. L'idée est donc de remplacer le travail fastidieux que représente l'étiquetage manuel d'un gros volume de données par celui consistant à concevoir et mettre en place un mécanisme d'étiquetage automatique. Dans ce projet, nous nous focaliserons sur l'algorithme SimCLR, celui-ci est plus particulièrement un algorithme semi-supervisé. C'est à dire : une première partie de notre modèle ne nécessite pas d'étiquettes pour l'apprentissage ; une deuxième partie nécessite les images labélisées mais d'un nombre bien moindre.

Chapitre 2

Etat de l'art du self surpervised learning

L'apprentissage non supervisé consiste à rechercher dans des données non étiquetées des groupes d'exemples partageant des caractéristiques communes. Dans la majorité des cas, ces méthodes sont liées au clustering. L'approche non supervisée ne nécessite pas d'étiqueter le jeu de données mais nécessite de nombreux exemples, des ressources de calcul et une fonction à définir pour décrire la différence entre les deux, ce qui n'est pas toujours facile.

Dans l'auto apprentissage, la méthode principale consiste à entraîner sur un jeu de données, par exemple des images, mais chacune d'entre elles est fournie en entrée dans sa forme originale et une version transformée. Ces transformations peuvent être de toute nature tel que le recadrage ou la rotation. Le modèle devra parvenir à minimiser la différence de prédiction entre le réseau interne qui avait en entrée l'image originale et qui, par conséquent, a une vision complète et inchangée de l'entrée, et la prédiction faite par le réseau qui a reçu l'image transformée

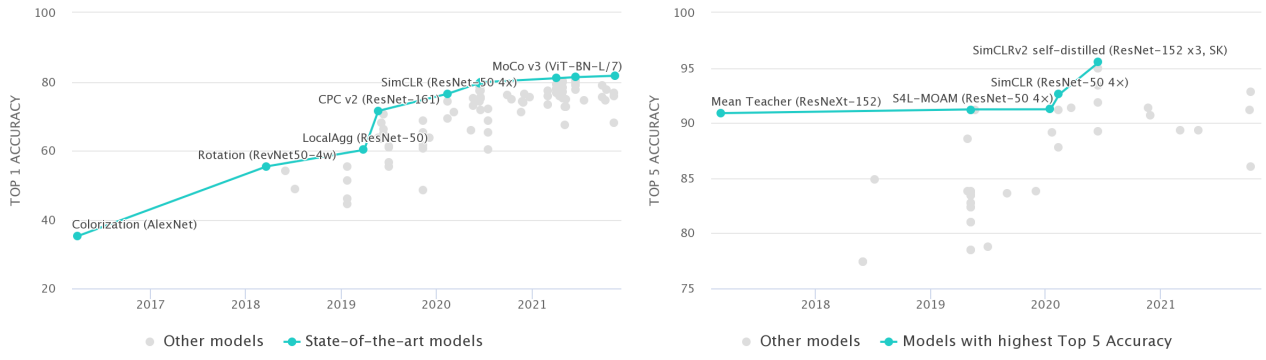

FIGURE 2.1 – Evolution des performances d'algorithmes de self supervised learning à gauche. Seulement les algorithmes contratifs à droite

Dernièrement, dans le traitement du langage naturel, les modèles Transformer ont obtenu beaucoup de succès. Des transformateurs comme Bert, T5 ont appliqué l'idée d'auto-supervision. Ils entraînent d'abord le modèle avec de grandes données non étiquetées, puis affinent le modèle avec quelques exemples de données étiquetées. Dans la classification des images sur ImageNet, ils sont aussi les algorithmes qui rencontrent les meilleures performances. Mais leur succès dépend aussi d'une demande considérable de données. Comme nous pouvons le voir, sur la Figure (2.1), les meilleurs sont des visions transformers.

Dans ce projet nous nous focaliserons sur une autre partie de l'apprentissage auto-supervisé, cette méthode s'appelle les apprentissages contrastifs. Apprentissage contrasté est une technique d'apprentissage automatique utilisée pour apprendre la caractéristiques générales d'un jeu de données sans étiquette en enseignant au modèle quels points de données sont similaires ou différents. L'apprentissage contrasté indique que pour toutes les paires positives x1 et x2, les sorties respectives f(x1) et f(x2) doivent être similaires et pour une entrée négative x3, f(x1) et f(x2) les deux doivent être dissemblables à f(x3). L'idée est représentée sur la figure (2.2)

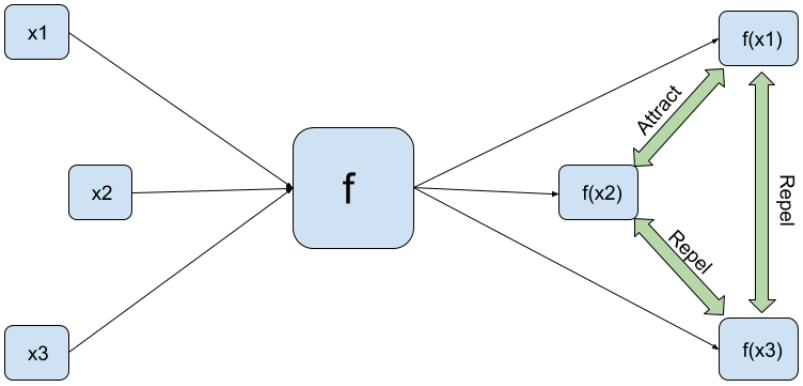

FIGURE 2.2 – Schéma d'apprentissage d'un algorithme contrastif

Nous allons maintenant nous concentrer sur le choix de nos données et sur la Baseline que nous utiliserons dans la suite de notre projet. Nous effecturons une présentation plus précise de l'algorithme SimCLR dans la suite

Chapitre 3

Choix des données et Baseline

Dans ce projet, nous choisissons de travailler sur le Dataset des chiens Stanford-Dogs. Ce choix permet d'avoir déjà fait une analyse exploiratoire, mais également, d'avoir une expérience sur les performances de certains algorithmes. Le nombre de photos du Dataset est de 20 000. Passons à la partie description de la séparation du dataset

Pour rentrer plus en détail, décrivons comment nous avons séparé les données pour avoir tous les dataset nécessaires. Dans un des modèles que nous détaillerons dans les prochaines parties, il était nécessaire d'avoir une base de données de photos de chiens sans labels. Cette base de données est composée de 12000 photos et elle nous servira à pré entrainer le SimCLR. Nous avions également besoin d'une petite partie du Dataset pour affiner deux modèles : pré entrainer l'un sur les Standford Dog et l'autre sur ImageNet. Cette partie nous servira également à entrainer notre baseline. J'ai décidé de prendre 10 % du Dataset Global, c'est à dire environ 2000 photos. Les 6000 photos restantes serviront de données de Test pour tester nos modèles sur des données inconnues.

	Pretrain	Train	Test
Nombre de photos	12000	1716	6864

La Baseline que nous allons utilisée est le modèle Inceptionv3, testée dans le projet précédent. Celui-ci était entrainé sur plus de 18 000 photos et avait comme Top 1 Accuracy plus de 70%. Pour l'adapter à notre contexte, je vais ainsi ré-entrainer un modèle Inceptionv3 avec seulement 2000 photos. Pour optimiser les performances et les hyperparamètres, j'ai utilisé un Hypertuner de Keras. Le Classifier est composé de 3 couches denses, d'un Drop Out.

Chapitre 4

Description de SimCLR

La façon la plus courante de modéliser cet auto-apprentissage consiste à prendre différentes forme d'une image en lui appliquant différentes augmentations et à passer les entrées modifiées à travers le modèle. Les images contiennent ainsi les mêmes informations visuelles mais ne se ressemblent pas, le modèle apprend que ces images contiennent les mêmes informations visuelles, c'est-à-dire le même objet. Cela conduit le modèle à apprendre une représentation similaire pour les mêmes objets.

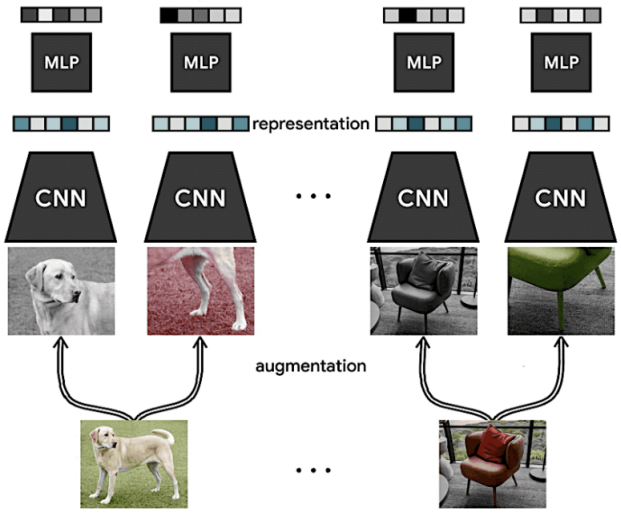

FIGURE 4.1 – Processus d'apprentissage de SimCLR

SimCLR créé des paires d'images à partir desquelles il apprend la similitude. Si nous saisissions la même image deux fois, il n'y aurait aucun effet d'apprentissage. Par conséquent, chaque paire d'images est créée en appliquant des augmentations ou des transformations à l'image. Ensuite, ces paires sont transmises à un réseau de neurones convolutifs pour créer une représentation de caractéristiques pour chacune des images. Dans l'article, les auteurs ont choisi d'utiliser l' architecture populaire ResNet pour leurs expériences.

Algorithm 1 SimCLR's main learning algorithm.

input: batch size N, constant τ, structure of $f, g, \mathcal{T}$.
for sampled minibatch $\{\boldsymbol{x}_k\}_{k=1}^{N}$ **do**
  for all $k \in \{1, \dots, N\}$ **do**
    draw two augmentation functions $t \sim \mathcal{T}, t' \sim \mathcal{T}$
    # the first augmentation
    $\tilde{\boldsymbol{x}}_{2k-1} = t(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k-1} = f(\tilde{\boldsymbol{x}}_{2k-1})$ # representation
    $\boldsymbol{z}_{2k-1} = g(\boldsymbol{h}_{2k-1})$ # projection
    # the second augmentation
    $\tilde{\boldsymbol{x}}_{2k} = t'(\boldsymbol{x}_k)$
    $\boldsymbol{h}_{2k} = f(\tilde{\boldsymbol{x}}_{2k})$ # representation
    $\boldsymbol{z}_{2k} = g(\boldsymbol{h}_{2k})$ # projection
  end for
  for all $i \in \{1, \dots, 2N\}$ and $j \in \{1, \dots, 2N\}$ **do**
    $s_{i,j} = \boldsymbol{z}_i^\top \boldsymbol{z}_j / (\|\boldsymbol{z}_i\| \|\boldsymbol{z}_j\|)$ # pairwise similarity
  end for
  define $\ell(i, j)$ **as** $\ell(i, j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(s_{i,k}/\tau)}$
  $\mathcal{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1, 2k) + \ell(2k, 2k-1)]$
  update networks f and g to minimize $\mathcal{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

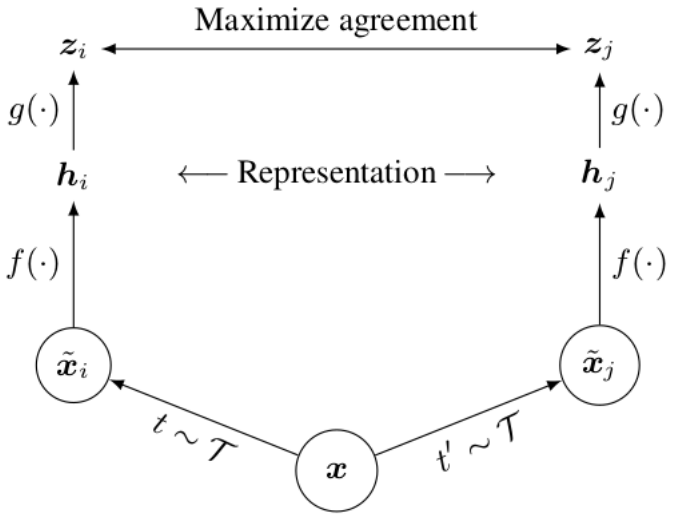

FIGURE 4.2 – τ est la fonction d'augmentation. La fonction f est le réseau de neuronnes ResNet. Le MLP est représenté par g

Une fois que la représentation vectorielle d'une image d'entrée est calculée par ResNet, cette sortie est transmise à une tête de projection pour un traitement ultérieur. Dans le papier, cette tête de projection est un MLP (Multi Layer Perceptron) avec une couche cachée. Ce MLP n'est utilisé que pendant l'apprentissage et l'affinement de la représentation des caractéristiques des images d'entrée. Une fois le calcul MLP terminé, le résultat se réserve comme entrée dans la fonction de perte. L'objectif d'apprentissage de SimCLR est de maximiser l'accord entre les différentes augmentations de la même image . Cela signifie que le modèle a essayé de minimiser la distance entre les images qui contiennent le même objet et de maximiser la distance entre les images qui contiennent des objets très différents.

Après la description de l'algortihme que nous utiliserons, passons à la partie modèlisation. Rappelons que le dataset que nous passerons au modèle est le Stanford Dogs. Notre Baseline sera Inceptionv3.

Chapitre 5

Modélisation

Dans cette partie, nous effectuerons les différentes modèlisations. Dans un premier temps, nous adapterons l'algorithme Inceptionv3 à notre problème et il servira de Baseline. Dans un deuxième temps, nous pré-entrainerons le modèle de simCLR sur nos données de Stanford Dogs, en entrainant sur 10 % des données étiquetées pour le classifier. Finalement, nous effectuerons un fine tuning du modèle simCLR fourni par les auteurs sur ces mêmes données étiquetées.

5.1 Inceptionv3

Pour l'entrainement d'Inceptionv3, nous utiliserons la même méthode que pour l'entraînement lors du projet précédent. C'est à dire que nous appliquerons un Tuner pour trouver les meilleurs paramètres sur notre Classifier. Nous figerons toutes les couches profondes d'Inceptionv3. Nous aurons ainsi 4 paramètres à faire varier. C'està dire le Learning Rate, deux couches denses et un Drop Out. Nous avons réalisé cet entrainement sur notre dataset de Train, c'est à dire environs 1700 photos. Ce choix de peu de photos est réalisé pour le comparer avec le peu de photos étiquetées dont nous avons besoin pour entrainer les modèles SimCLR.

	Inceptionv3
Accuracy Top 1	4.1

Les performances du modèle sont reportées dans le Tableau suivant. Nous obtenons 3% d'accuracy sur les 120 races de chiens. Nous pouvons maintenant passer à nos modèles de SimCLR

5.2 SimCLR avec pré-entrainement sur les photos de chiens

Comme nous l'avons expliqué dans la partie portant sur SimCLR, dans un premier temps, SimCLR nécessite un pré-entraînement à réaliser sur les photos pour qu'il puisse bien distinguer les différentes classes de photos. Nous décidons de réaliser ce pré-entrainement grâce au répertoire git mis à disposition par les

auteurs de l'article. Nous pré-entraînons le modèle sur environs 12 000 photos, ce qui n'est pas beaucoup et ce qui risque de limiter les performances du modèle. Nous faisons varier les paramètres du modèle. Mais par limite de temps, ces recherches n'ont pas pu être approfondies jusqu'à l'optimisation.

Une fois notre modèle pré-entrainé sans les labels, nous entraînons la tête linéaire avec les 1700 photos labélisées et nous regardons les performances reportées dans le tableau ci dessous. Comme nous pouvons le voir, les performances de notre modèle dépassent déjà celle de Inceptionv3, malgré que notre pré-entraînement ne soit pas optimal et le peu de photos sur lesquelles nous l'avons entrainé.

	InceptionV3	SimCLRv1
Accuracy Top 1	4.1	6,8

Pour constater des performances que nous pourrions atteindre, nous récuperons le modèle pré-entraîné des auteurs de l'article. Ce modèle a été entrainé sur ImageNet et nous cherchons ainsi à re-entraîner la tête linéaire pour avoir une idée des performances minimales que nous pourrrions atteindre une fois tout le modèle bien otpimisé.

5.3 SimCLR avec pre entrainement sur ImageNet

Cette fois-ci, il n'est pas nécessaire de pré-entraîner le modèle. Celui-ci a été entrainé sur ImageNet par les auteurs et très bien optimisé. Nous avons simplement à entraîner la tête linéaire sur les 1700 photos étiquetées. Les performances du modèle sont reportées dans le tableau suivant.

	Inceptionv3	SimCLRv1	SimCLRv2
Accuracy Top 1	4.1	6.8	62,5

Comme nous l'observons les performances de celui-ci sont bien supérieures à celles que nous avons obtenu. Ceci n'est pas très étonnant car le modèle est mieux optimisé et a été entrainé à bien mieux séparer les photos que ne l'a été notre modèle maison. Cette application nous permet de voir que les performances du modèle SimCLR, s'il avait été optimisé pourrait être bien supérieur aux modèles Inceptionv3. Elles pourraient atteindre celles d'Inceptionv3 entraîné classiquement.

Chapitre 6

Conclusion

Dans ce projet, nous avons d'abord dû trouver une problèmatique liée à des développements récents sur le machine learning. Cela nous a permit d'apprendre à nous renseigner sur les nouveaux développement et nous a entrainé à le mettre en place.

Pour prouver l'efficacité de cette nouvelle méthode, nous avons choisi de les mettre en place sur un dataset que nous connaissions déjà par les projets précédent. Ceci nous a permis d'utiliser les résultats que nous avions obtenus et de les utiliser comme Basline de comparaison. Nous avons pu mettre en place une méthode pour prouver l'efficacité de ce nouvel algorithme puisque les performances dans le contexte que nous avons défini sont meilleurs.

Ainsi, les modèles contrastif de self learning peuvent permettre de faire de la classification sans avoir nécessairement besoin des étiquettes ou alors besoin d'un moins grand nombre d'étiquettes pour pouvoir être performant. Cela permet d'éviter le travail fastidieux et humain d'étiqueter des photos pour avoir des algorihmes de Classification fficace. Même si nos résultats ne dépassent pas les résultats que nous avions obtenus lors du projet sur les Stanford Dogs, nous avons pû montrer que cette méthode était plus efficace vu notre nombre de données. Et comme nous pouvons le voir dans la littérature, lorsque le modèle est bien optimisé, les résultats sont de plus en plus performants.