Projet de Deep Learning
« Differentially Private Releasing via Deep Generative Model »

Alexandre Huat
INSA Rouen Normandie
Master Science des Données

21 mars 2018

1 Résumé d'article

Supposons un groupe de clients et un prestateur de services informatiques collectant et analysant leurs données (*e.g.* classification d'images). Au cours des tâches d'analyses, le prestateur est amené à traiter des données sensibles. Afin de respecter la vie privée de ses clients, il doit trouver un moyen de traiter efficacement ces données tout en conservant leur confidentialité. C'est à cette problématique que répondent Zhang, Ji et Wang [1] par l'architecture dp-GAN de l'article résumé ici.

La Figure 1 illustre le rôle de dp-GAN, qui est de générer des données synthétiques mais sémantiquement riches, *i.e.* suffisamment représentatives des clients, sans violation de leur vie privée. En introduction, les auteurs rappellent les défis à relever en apprentissage sous contrainte de confidentialité et présentent les apports de dp-GAN. En plus de sa capacité à générer une infinité de données, dp-GAN garantit l'anonymisation des données réelles par respect du principe de « confidentialité différentielle » [1]. Pour ce faire, dp-GAN applique au réseau adverse génératif de Wisserstein (WGAN) amélioré des mécanismes d'anonymisation à l'état de l'art, alors optimisés pour gagner en stabilité et en scalabilité.

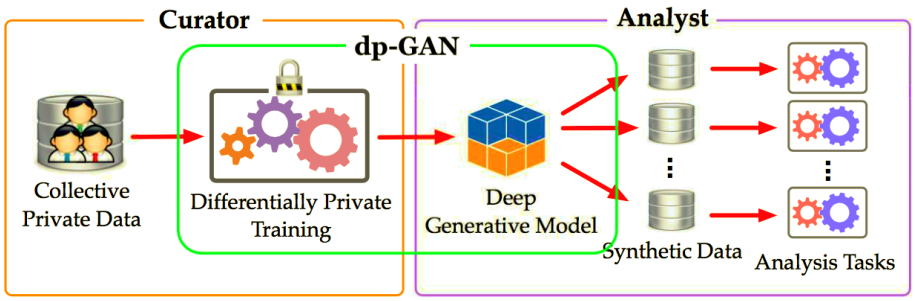

Figure 1 La place de dp-GAN dans la chaîne de traitement des données confidentielles. Le « *curator* » est l'entité qui anonymise les données pour l'*analyst*.

En Section 2, Zhang *et al.* font un rappel théorique sur les GAN et justifient leur utilisation du WGAN amélioré par une plus grande stabilité et un plus court temps d'apprentissage que le GAN originel. Ils font également la définition formelle de la confidentialité différentielle et citent des propriétés associées dont bénéficient dp-GAN. La Section 3 présente dp-GAN dans sa version basique et fournit son algorithme d'apprentissage (Algorithme 1). Pour assurer sa confidentialité, à chaque mise-à-jour, l'algorithme bruite le gradient du discriminateur (bruitage gaussien et seuillage), à partir duquel un pirate pourrait autrement reconstruire les données privées. Cette technique est communément utilisée dans la littérature. Une preuve théorique du niveau de confidentialité différentielle atteint par l'algorithme est également apportée.

1. Synthétiquement, la confidentialité différentielle mesure la capacité d'un tiers à déduire des données privées des résultats d'un algorithme ; *cf.* exemple à https://fr.wikipedia.org/wiki/Confidentialité_différentielle#Formalisation.

Néanmoins, cette version de dp-GAN possède trois inconvénients : (i) elle génère des données de faible qualité ; (ii) elle converge moins rapidement que le GAN non-confidentiel, voire diverge ; (iii) elle est rigide et n'exploite aucune ressource bonus, *e.g.* des données publiques. Pour palier ces défauts, la version avancée de dp-GAN implémente : (i) un regroupement des paramètres du réseau pour un réglage fin et spécifique de leurs bruits respectifs ; (ii) un seuillage adaptatif du gradient, qui évolue au cours des itérations ; (iii) une initialisation des paramètres du réseau par pré-apprentissage sur les données publiques disponibles. Ces améliorations boostent la vitesse de convergence et la confidentialité de dp-GAN. La Section 4 détaille l'algorithme de cette version avancée (Algorithme 3).

S'en suit un rapport d'expériences sur trois bases célèbres et libres d'accès : MNIST, CelebA et LSUN. LSUN est ensuite divisée en deux bases, l'une labellisée (LSUN-L) et l'autre non-labellisée (LSUN-U). Les expériences sont réalisées avec TensorFlow (TF) mais le code n'est pas partagé par ses auteurs. Certains paramètres de tests sont renseignés mais l'architecture des réseaux générateurs et discriminateurs ne le sont pas. Ainsi, la Section 5 propose une évaluation qualitative et quantitative des performances du système. De mon point de vue, les images générées par dp-GAN, quelle que soit la base, sont assez vraisemblables ; MNIST en particulier est très bien simulée. Dans leur deux premières expériences, Zhang *et al.* comparent quantitativement la qualité des données générées par dp-GAN aux données réelles et à celles générées par le GAN non-confidentiel. dp-GAN performe légèrement moins bien pour les données labellisées comme pour les non-labellisées. Dans leur troisième expérience, les auteurs comparent les performances atteintes en classification sur LSUN-L après apprentissage sur : (i) les données réelles seules ; (ii) les données réelles jointes aux données synthétisées par un GAN non-confidentiel ; (iii) les données réelles jointes aux données synthétisées par dp-GAN. Il en ressort que l'apprentissage avec les données synthétiques permet systématiquement une diminution des taux d'erreurs (jusqu'à -3.3 % pour dp-GAN, contre -7.7 % pour le GAN non-confidentiel). Quatrièmement, en terme de score d'Inception et de Jensen-Schannon, une ultime expérimentation valide l'efficacité des stratégies d'optimisations dont bénéficie dp-GAN avancé.

Dernièrement, les auteurs consacrent une section aux travaux similaires de la littérature. Puis, ils concluent en rappelant l'intérêt et les améliorations apportées par dp-GAN et, enfin, ouvrent la discussion sur la limite que l'architecture n'a été testée que sur des images – une évaluation sur d'autres types de données (*e.g.* texte) étant bienvenue.

2 Implémentation

TODO Un gros rapport, complet, avec explication du code, difficulté, est-ce que j'ai repris du début, mes résultats expérimentaux, etc.

Cette section détaille mon implémentation de dp-GAN, les difficultés rencontrées et leurs solutions possibles (possiblement non-implémentées) ainsi que les résultats obtenus.

Le code du projet est disponible sur Github à : `https://github.com/alexandrehuat/dp-gan`.

2.1 Objectifs

Le véritable défi de ce projet, et en accord avec l'objet du cours de Deep Learning, est l'implémentation de dp-GAN basique. En effet, celle-ci demande une prise en main de WGAN amélioré et des calculs de confidentialité différentielle [2], [3]. En comparaison, dp-GAN avancé ne consiste qu'en l'enrichissement de dp-GAN basique d'un apprentissage sur des données publiques et d'un *clustering* hiérarchique des paramètres du réseau. Or, comptant ma formation en apprentissage statistique, ces opérations ne présentent pas de plus-value pédagogique. Sachant le temps allouable au projet, il m'est apparu raisonnable de me concentrer sur l'implémentation de dp-GAN basique. Quant aux données d'expérimentations, j'ai choisi d'utiliser MNIST pour faciliter l'apprentissage et l'interprétabilité des résultats.

Algorithm 1: Basic dp-GAN

Input: n - number of samples; λ - coefficient of gradient penalty; n_{critic} - number of critic iterations per generator iteration; n_{param} - number of discriminator's parameters; m - batch size; $(\alpha, \beta_1, \beta_2)$ - Adam hyper-parameters; C - gradient clipping bound; σ - noise scale; (ϵ_0, δ_0) - total privacy budget

Output: differentially private generator G

while θ *has not converge* **do**
  for $t = 1, \cdots, n_{critic}$ **do**
    for $i = 1, \cdots, m$ **do**
      sample $x \sim p_{\text{data}}, z \sim p_z, \rho \sim \mathcal{U}[0,1]$;
      $\hat{x} \leftarrow \rho x + (1-\rho) G(z)$;
      $\ell^{(i)} \leftarrow D(G(z)) - D(x) + \lambda \left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1 \right)^2$;
      // computing discriminator's gradients
      $g^{(i)} \leftarrow \nabla_w \ell^{(i)}$;
      // clipping and perturbation
        $(\xi \sim \mathcal{N}(0, (\sigma C)^2 \mathcal{I}))$
      $g^{(i)} \leftarrow g^{(i)} / \max(1, ||g^{(i)}||_2 / C) + \xi$;
    // updating privacy accountant
    update $\mathcal{A}$ with $(\sigma, m, n_{\text{param}})$;
    // updating discriminator
    $w \leftarrow \text{Adam}\left(\frac{1}{m} \sum_{i=1}^{m} g^{(i)}, w, \alpha, \beta_1, \beta_2 \right)$;
  sample $\{z^{(i)}\}_{i=1}^{m} \sim p_z$;
  // updating generator
  $\theta \leftarrow \text{Adam}\left(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D(G(z^{(i)})), \theta, \alpha, \beta_1, \beta_2 \right)$;
  // computing cumulative privacy loss
  $\delta \leftarrow$ query $\mathcal{A}$ with ϵ_0;
  if $\delta > \delta_0$ **then** break;
return G

2.2 Algorithme, technologie et organisation

Comme dit en section 1, l'algorithme d'apprentissage de dp-GAN basique (Algorithme 1) est celui de WGAN amélioré avec un bruitage et seuillage de gradients. Étant donné l'avénement de TF dans la communauté du Deep Learning ainsi que l'utilisation de cette librairie par Zhang, Ji et Wang [1], j'ai choisi de réaliser toute la partie bas niveau du projet (apprentissage de dp-GAN) en TF et d'utiliser Keras pour les opérations de plus haut niveau (création, sauvegarde et évaluation des modèles). L'utilisation de TF était nécessaire pour brouiller le gradient (ligne 8), en recherche de confidentialité. Sans ces opérations, le projet aurait essentiellement été une implémentation de WGAN amélioré. Cette dernière aurait été totalement réalisable en Keras par alternance d'apprentissage du générateur (G) et du discriminateur (D) en gelant les poids de G quand D apprend, resp. de D quand G apprend.

De plus, connaissant Keras par de précédents cours, projets personnels ou le PIC Becquerel 2017, un projet en Keras pur m'aurait permis de me concentrer sur les tests ou améliorations de la méthode codée. À l'inverse, je ne connaissais pas TF avant ce projet. Je dédiai donc environ deux tiers de mes efforts au développement de l'Algorithme 1 en TF. Puis, je consacrai le temps restant au pré-apprentissage et tests des modèles. Je reviendrai sur ces points en subsection 2.3 et subsection 2.4.

Enfin, je dois préciser qu'il ne m'a pas été possible d'implémenter la partie « vérification de la confidentialité » de dp-GAN (lignes 9 et 13). En effet, les auteurs donnaient trop peu d'explications sur ces opérations ; il m'aurait fallu lire [2] et [3] pour les intégrer. En outre, je l'ai omise, car cette partie sortait du cadre Deep Learning, et il me fallait tester mon code dans le temps imparti.

2.3 Apprentissage

préapprentissage AE 50 epochs, test mse = 0.0297 CNN 2 epochs, test acc = 98.57 %
validation = 10 %

DIFFICULTÉS : * première impl avec "with tf.Session()" : app avec tf -> pred avec keras -> "FailedPreconditionError (see above for traceback) : Attempting to use uninitialized value is$_real/bias$" $solution$: $self.tf_session!rfrencejamaisperduedonccalculstjrspossible$

DPGAN TRAINING AVEC 1000 EPOCHS, LAMDA 10 ALPHA 0002 BETA1 09 BETA2 05 Discriminator's mean overestimation of X$_test$: $-0.4642 Discriminator'smeanoverestimationofZ$: $0.6233 Discriminator'smeanoverestimationofG(z)$: 0.5258

e 1000 l 10 a 0.001 b1 0.9 b2 0.999 C 0.5 s 1 Discriminator's mean overestimation of X$_test$: $-0.5864 Discriminator'smeanoverestimationofZ$: $0.4100 Discriminator'smeanoverestimationofG(z)$: 0.3943

2.4 Évaluation

2.5 Interfaçage de TF et Keras

De manière générale, l'interfaçage de Keras et TF est fluide, presque transparent. Cependant, j'ai rencontré une difficulté lorsque je souhaitais évaluer les réseaux via Keras après leur apprentissage avec TF. EN effet, j'obtenais une erreur du type :

```
FailedPreconditionError: Attempting to use uninitialized value <layer_name>/bias
```

Cette erreur était causée par la perte de la référence des poids des réseaux après l'apprentissage. Ceci car les poids dépendent de la session TF qui exécute leur mise à jour, et que cette session était éphémère dans ma première implémentation. Ainsi, j'ai résolu ce problème en pérénisant la session sous la forme d'un attribut de `DPGAN` (*cf.* Listing 2) plutôt que d'utiliser une session locale à la fonction d'apprentissage (*cf.* Listing 1).

Références

[1] X. Zhang, S. Ji et T. Wang, « Differentially Private Releasing via Deep Generative Model », *ArXiv e-prints*, jan. 2018. arXiv : 1801.01594 [cs.CR].

Listing 1 Avant : Gestion des sessions TF avec erreur

```
def BasicDPGAN(DPGAN):
    def train(...):
        ...  # Préparation des calculs
        with tf.Session() as sess:
            <results> = sess.run(<training>)  # Exécution des calculs
        return <results>  # La référence est supprimée à ce moment
```

Listing 2 Après : Gestion des sessions TF sans erreur

```
def BasicDPGAN(DPGAN):
    def train(...):
        ...  # Préparation des calculs
        <results> = self.tf_session.run(<training>)  # Exécution des calculs
        self.tf_session.close()
        return <results>
```

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar et L. Zhang, « Deep Learning with Differential Privacy », in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, sér. CCS '16, Vienna, Austria : ACM, 2016, p. 308–318, ISBN : 978-1-4503-4139-4. DOI : 10.1145/2976749.2978318.

[3] F. D. McSherry, « Privacy Integrated Queries : An Extensible Platform for Privacy-preserving Data Analysis », in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, sér. SIGMOD '09, Providence, Rhode Island, USA : ACM, 2009, p. 19–30, ISBN : 978-1-60558-551-2. DOI : 10.1145/1559845.1559850.