GENERATIVE ADVERSARIAL NETS

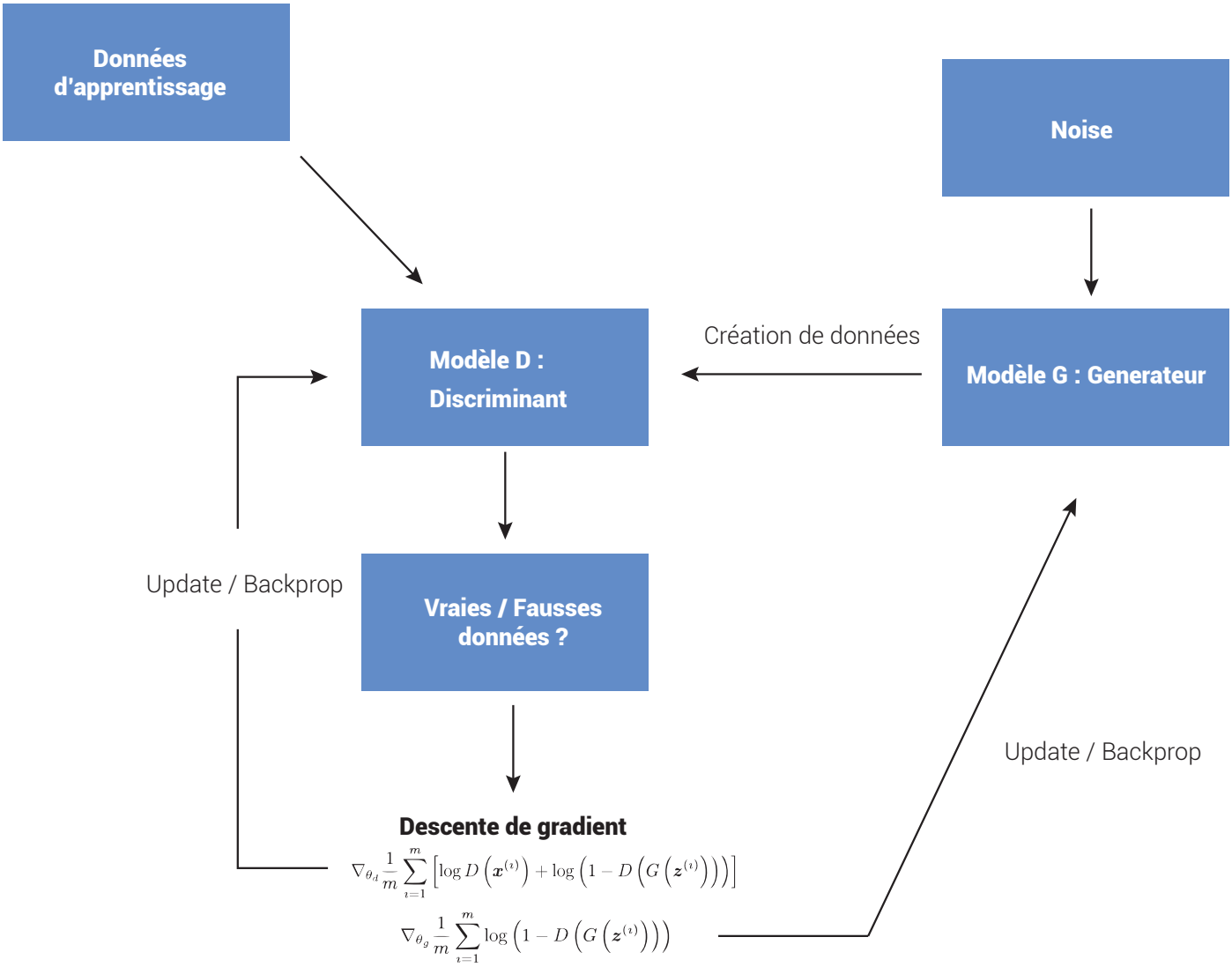
Basile Guerrapin & Timothée Poulain

PROBLEMATIQUE & CONTEXTE

L'article que nous avons choisi d'étudier est « Generative Adversarial Nets ». Cette publication peut paraître un peu ancienne dans le monde du Machine Learning tant les avancées sont rapides. Elle date du 10 juin 2014. L'un des auteurs de cet article, Ian Goodfellow est très influent dans le domaine de l'intelligence artificielle. Il est un membre clé aujourd'hui de la société OpenAI, fondée par Elon Munsk.

Dans le domaine du deep-learning, les modèles architecturaux tels que les modèles discriminants remportent un franc succès. Ils ont la possibilité d'être utilisés dans de multiples dimensions afin de classifier de grande quantité de données parmis une collection d'étiquettes. Au sein des modèles discriminants, les réseaux de neurones sont aisément utilisables car ils sont différentiables donc supportent bien la rétro-propagation du gradient.
Dans ce document, les auteurs décident de mettre en avant un autre modèle architectural que le modèle utilisé habituellement, le modèle génératif. Ce modèle est une approche très prometteuse dans le domaine du deep-learning. Durant de nombreuses années, les modèles génératifs n'ont pas eu un développement heureux. En effet, les chercheurs se heurtent à des problématiques difficilement solubles liées à l'estimation de vraissemblance.
« Generative Adversarial Nets » est un des premiers articles à mettre en lumière une facilité d'utiliser des modèles génératifs.
Pour le modèle génératif, nous nous positionnons dans un cadre d'apprentissage non supervisé.
Afin de crée ce modèle, nous devons collecter un certain nombre de données afin de générer au travers d'un réseau de neurones des « échantillons du modèle ».
Diverses problématiques se dégagent de cette approche :
- Comment met-on en oeuvre un modèle génératif ?
- Comment ajuster les différents paramètres de ce réseau afin d'obtenir une distribution d'échantillons de plus en plus réaliste ?

Dans cet article, les auteurs ont choisi de mettre en avant non pas un mais deux modèles : un modèle génératif et un modèle discriminatif pour répondre à ces problématiques.
L'objectif du modèle discriminant est de trouver quelles sont les données qui proviennent des données réelles et quelles sont les données provenant du générateur.
L'intérêt est d'avoir deux modèles concurrents de réseaux de neurones. Le générateur G reçoit du bruit en entrée afin de produire des données.
Le discriminateur reçoit en même temps des echantillons de la base d'apprentissage et en même temps des données généré par le modèle G.
Le générateur apprend de plus en plus à générer des données réalistes. Ce qui permet à la fin au discriminateur de distinguer d'une manière singulière les différences entre les données générées et les échantillons.
Ce qu'il faut mettre en avant est le principe de concurrence. En effet, l'objectif final est que les données générées par G soient indiscernables des échantillons. Pour cela, les auteurs utilisent l'algorithme Minimax que l'on regarde de plus près dans la suite de ce projet.
Cette architecture met en avant le fait que l'on peut retro-propager aussi bien le gradient du modèle discriminant que le gradient du modèle génératif. Celui-ci pourra alors générer des données afin de tromper le discriminant.



GAN'S MATHS

//L'algorithme du Min-Max :
Pour notre problématique, nous utilisons l'algorithme du MinMax afin d'entrainer le modèle.

$$\max_D V(D,G) \rightarrow \min_G V(D,G) \rightarrow \max_D V(D,G) \rightarrow \cdots$$

Ci-dessous la fonction Minimax adaptée au modèle que nous allons traiter.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Finalement, l'auteur préconise d'effectuer l'astuce suivante car elle fournit des gradients plus forts au début de l'apprentissage.

$$\min_G \quad \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

$$\Leftrightarrow \min_G \quad \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

$$\Leftrightarrow \max_G \quad \mathbb{E}_{z\sim p_z(z)}[\log(D(G(z)))]$$

PLANS D'EXPERIENCES

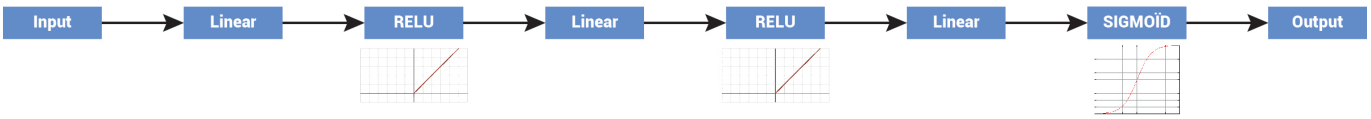
Nous avons choisi d'essayer trois différents plans d'experience. Le premier est basé sur l'approximation d'une distribution gaussienne. Le deuxième plan d'expérience s'effectue sur l'approximation des données MNIST et nous proposons une ouverture sur les réseaux convolutifs dans le troisième plan d'experience.

// Plan d'expérience « Distribution Gausienne »

Au départ, nous testerons deux architecture différentes. La première (ci-dessous) est issu de [2] et la seconde (présentée dans le plan d'expérience suivant) provient du papier original [1]. En effet, l'architecture présentée dans [2] apporte quelques modifications que nous jugeons intéressantes à investiguer dans le cadre de notre démarche : en particulier un modèle G moins profond que le modèle D.

Architecture Modèle G :

Architecture Modèle D :



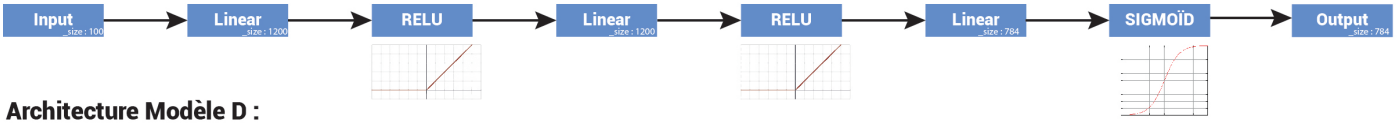
Le critère utilisé est BCECriterion : Binary Cross Entropy

$$Loss(o,t) = -\frac{1}{n}\sum_i (t[i] \times \log(o[i])) - \frac{1}{n}\sum_i ((1 - t[i]) \times \log(1 - o[i])) \qquad \text{avec t[i] = 0 ou 1}$$

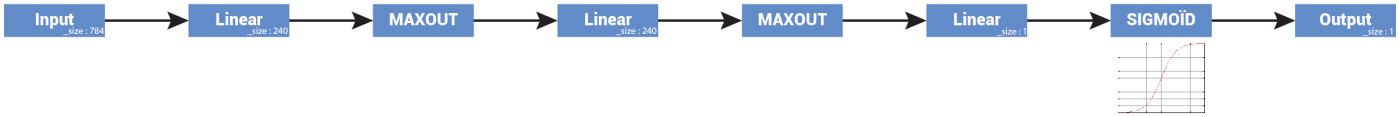
Dans un premier temps, nous utiliserons une descente de gradient stochastique, sans mini-batchs. Dans un second temps, nous ajusterons la taille des mini-batchs. Nous avons lu dans [2] que la taille du mini-batch est un hyperparamètre important à régler (l'auteur précise que la valeur maximale est 16).

// Plan d'expérience « MNIST »

Architecture Modèle G :



Architecture Modèle D :



Dans l'objectif de comparer et présenter nos résultats, nous expérimenterons ensuite les algorithmes sur un dataset d'images (MNIST). Nous pourrons à nouveau comparer les deux architectures.

// Plan d'expérience (Facultatif)

Nous souhaitons investiguer l'utilisation d'architecture convolutionnelle. En effet, de nombreux articles traitent de ce problème.
Ce plan d'expérience est facultatif car nous le mettrons en oeuvre seulement si nos capacités matérielles sont suffisantes. Pour cela, nous pourrons utiliser le dataset Cifar-10. Ainsi que les différents articles énoncés en bibliographie [3] [4] [5].

Le framework Generative Adverserial Network possède de nombreux avantages :
- Back-propagation facilitée, non utilisation des chaines de Markov, les données ne sont pas directement copiées dans le générateur mais proviennent des gradients du discriminateurs.
Toutefois, on identifie un inconvénient, la representation de $p_g(x)$ n'est pas explicite.
Il faut aussi bien faire attention à la synchronisation des deux modèles pendant l'entrainement.

// Bibliographie

_ [1] Generative Adversarial Nets, « https://arxiv.org/pdf/1406.2661v1.pdf»

_ [2] An introduction to Generative Adversarial Networks (with code in TensorFlow),
« http://blog.aylien.com/introduction-generative-adversarial-networks-code-tensorflow/»

_ [3] The Eyescream Project, « http://soumith.ch/eyescream/»

_ [4] Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks,
« https://arxiv.org/pdf/1506.05751v1.pdf»

_ [5] Generating Faces with Torch, « http://torch.ch/blog/2015/11/13/gan.html»