Utilité des réseaux neuronaux dans la cryptographie

BLANC Swan & LE BRAS Clément

La présentation suivante est basée sur l'article : "*Learning to Protect Communications with Adversarial Neural Cryptography"* [1].

Introduction

Dans cet article, il s'agit d'expliquer un exemple d'utilisation de réseaux de neurones pour réaliser la cryptographie.
Généralement, les réseaux neuronaux ne sont pas conçus pour être très efficaces dans ce domaine. Par exemple, le plus simple d'entre eux n'est pas en mesure de reproduire la fonction XOR qui est la base de nombreux algorithmes cryptographiques.
Néanmoins, il est possible de démontrer que les réseaux de neurones peuvent servir à protéger la confidentialité de leurs données à l'égard d'autres réseaux. Ils permettent ainsi la découverte de nouvelles formes de cryptage et de décryptage, plus particulièrement l'utilisation de clefs secrètes sans imposer d'algorithme de chiffrement spécifique.

Il s'agit à présent, d'expliquer comment deux réseaux de neurones peuvent apprendre à communiquer entre eux de façon sécurisée.

Scénario

Alice et Bob veulent communiquer clairement, mais sans qu'Eve les comprenne. Celle-ci ne peut qu' intercepter les communications. Elle ne peut pas ajouter ou modifier des messages.

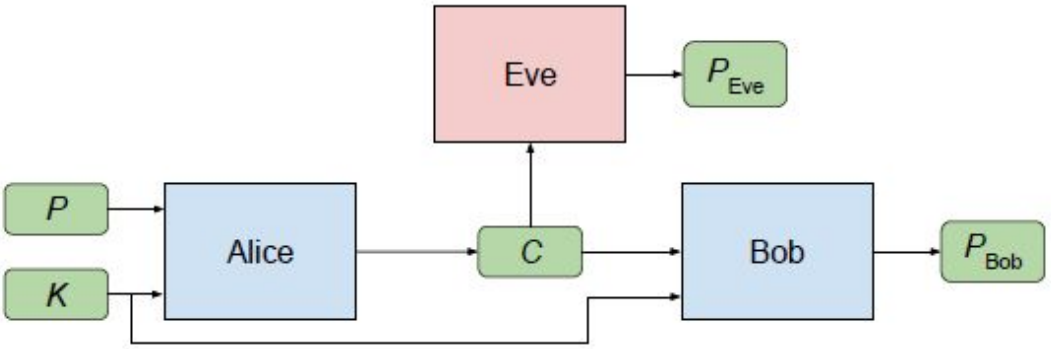

figure 1: Scénario

La figure 1 montre que :

-Alice envoie un message P à Bob (P est une entrée du réseau neuronal Alice).
-Alice crypte le message qui devient C.

Les deux réseaux neuronaux Eve et Bob reçoivent le message C et tentent de retrouver le message P. P_{Bob} représente le message décrypté par Bob et P_{Eve} représente le message décrypté par Eve.
Alice et Bob se partagent une clé K mais Eve n'y a pas accès.

Les réseaux de neurones auront leurs propres paramètres θ_A, θ_B et θ_E. Pour information θ_A et θ_B n'ont pas besoin d'être égaux. Il en est de même pour la fonction de cryptage et de décryptage, même si Alice et Bob ont la même structure. Les valeurs K et P sont des listes de flottants bornées entre -1 et 1.

Pour des raisons de simplicité, l'étape de la génération de la clé à partir d'une seed est ignorée.

Objectif du scénario

Comme décrit précédemment, Alice et Bob communiquent clairement entre eux. Il est nécessaire donc de minimiser le nombre d'erreurs entre P et P_{Bob}.

Au lieu d'entraîner Alice et Bob à mettre en place séparément certains systèmes de chiffrement connus, il est question de les entraîner conjointement à communiquer ensemble. Leur objectif est de battre Eve, c'est à dire d'augmenter le nombre

d'erreurs entre P et P_{Eve}. Ce type de mise en compétition se nomme GAN: "Générative Adversarial Network" [2]. Grâce à celle-ci, Alice et Bob vont améliorer la robustesse de leur communication.

On note:

- $A(\theta_A, P, K)$ le message crypté d'Alice: C
- $B(\theta_B, C, K)$ le message décrypté de Bob: P_{Bob}
- $E(\theta_E, C)$ le message décrypté d'Eve: P_{Eve}
- d la distance entre 2 messages $d(P, P') = \sum_{i=1}^{N} |P_i - P'_i$ où N est la longueur du message
- $L_E(\theta_A, \theta_E, P, K) = d(P, E(\theta_E, A(\theta_A, P, K)))$ est la Fonction Objectif d' Eve.
- $O_E(\theta_A) = argmin_{\theta_E}(L_E(\theta_A, \theta_E, P.K))$ Son objectif est de minimiser l'erreur entre P et P_E.
- $L_B(\theta_A, \theta_B, P, K) = d(P, B(\theta_B, A(\theta_A, P, K), K))$ est la Fonction Objectif de Bob.
- $L_{AB}(\theta_A, \theta_B, P, K) = L_B(\theta_A, \theta_B, P, K) - L_E(\theta_A, O_E(\theta_A))$ est la Fonction Objectif de Alice et Bob. Celle-ci montre qu' Alice et Bob veulent minimiser le nombre d'erreurs de P_{Bob} et maximiser le nombre d'erreurs de P_{Eve}
- $(O_A, O_B) = argmin_{(\theta_A, \theta_B)}(L_{AB}(\theta_A, \theta_B, P, K))$ est la fonction optimale pour minimiser la fonction L_{AB}.

"Optimal" signifie faire converger les réseaux de neurones vers une solution optimale. Par exemple, si la longueur de la clé est égale à celle du texte, la solution optimale serait qu'Alice et Bob appliquent un XOR, mais ils peuvent en plus permuter des valeurs identiques dans la clé.

Scénario d'entraînement

Voici à présent la partie entraînement du réseau de neurone dont l'objectif est de converger Alice et Bob vers (O_A, O_B) et Eve vers O_E.

L'idée générale de l'entraînement est d'alterner [2] une phase d'entraînement d'Alice et Bob avec une phase d'entraînement d'Eve. Cette alternance permet, dans un premier temps, d'entraîner Alice et Bob à communiquer entre eux puis, dans un second temps, d'entraîner Eve à retrouver le message d'Alice.

L'entraînement se termine une fois qu'Alice et Bob ont trouvé leur protocole de communication et qu'après plusieurs phases successives d'entraînement, Eve ne soit pas parvenue à décrypter leur message.

Il est important de préciser, qu'au début de chaque entraînement, il est nécessaire d' initialiser aléatoirement les poids d'Alice, Bob et Eve.

Architecture des réseaux de neurones

L'architecture choisie se veut la plus générique possible. Elle doit permettre de mélanger des données et d'appliquer des fonctions telles que des XOR. L'architecture se nomme "mix & transform".

Premièrement il existe un "Fully-Connected layer"[3]:

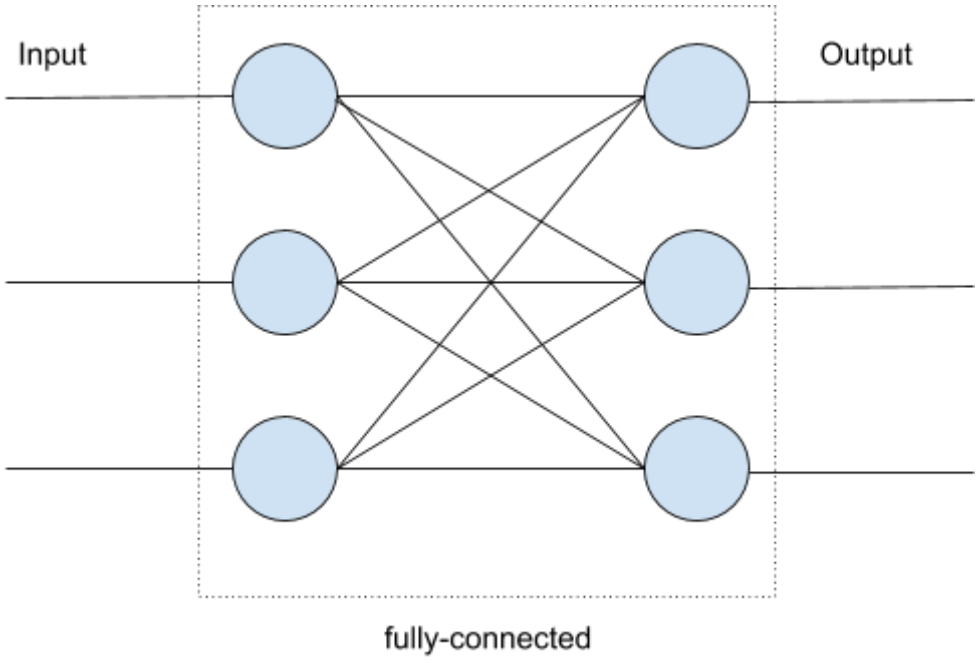

figure 2: fully-connected layer (FC)

Le nombre d'entrées est égale au nombre de sorties. La clé et le texte clair sont introduits dans cette couche. Une "Fully-Connected layer" est choisie car chaque sortie peut être une combinaison linéaire de toutes les entrées. De plus, cette couche peut permettre de mixer la clé avec le texte clair.

Elle est suivie d'un "Convolutional Neural Network"[3] qui permet d'appliquer des fonctions pour grouper des bits préalablement mixés par la couche précédente, sans spécification à priori de ce que cette fonction devrait être.

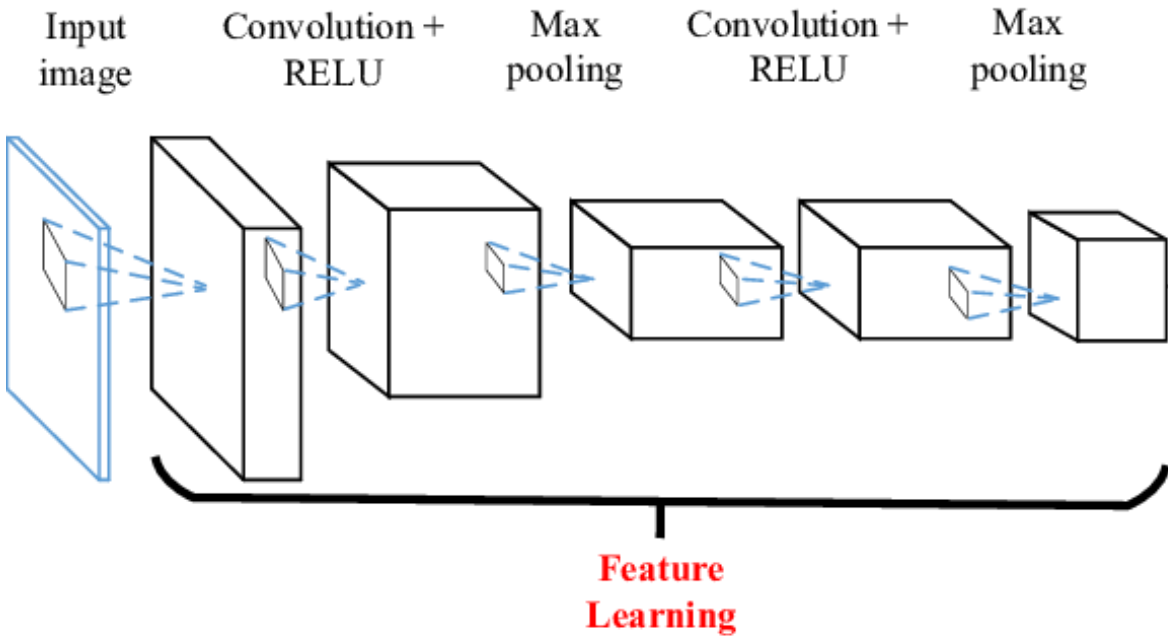

figure 3: Ce qui est représenté sur cette figure est un CNN 2 dimension

Expérimentations

Voici à présent la mise en oeuvre de la procédure.

Structure des réseaux de neurones

Un nombre de clés et de textes clairs égal à la valeur du minibatch (ici 4096) et de longueur N bits aléatoires (16, 32 ou 64) a été généré. La clé n'est pas ré-utilisée délibérément mais pourrait réapparaître puisqu'elle est choisie aléatoirement.

L'expérience a été réalisée avec TensorFlow.

Alice prend en entrée la clé et le texte clair. Leurs valeurs sont comprises entre -1 et 1. Ce vecteur de taille $2N$ va être traité par le “Fully-Connected Layer” de $2N$ * $2N$ de dimension, puis envoyé vers 4 “Convolutional Neural Networks” qui vont appliquer une convolution de dimension 1. En sortie, Alice réduit la valeur à N éléments.

Bob est identique à Alice mais Eve ne prend uniquement le texte chiffré en entrée. La “Fully-Connected Layer” est cette fois si de dimension $N * 2N$.

Entraînement des réseaux de neurones

A chaque phase d’entraînement, Alice et Bob seront soumis à 4096 valeurs (1 minibatch) et Eve au double, c’est à dire 8192 valeurs (2 minibatchs), afin de donner un avantage à l’adversaire. Il est nécessaire de rappeler l’alternance entre l’entraînement de Alice et Bob et celui d’Eve. Il a été décidé de commencer avec l’entraînement d’Alice et Bob.

Les Fonctions Objectifs

Eve a pour Fonction Objectif L_E. Son but est de minimiser l’erreur entre P et P_E.

Alice et Bob ont pour Fonction Objectif L_{AB} dont le but est de minimiser le nombre d’erreurs de P_{Bob} et de maximiser le nombre d’erreurs de P_{Eve}.

Evaluation post-entraînement

Une fois que la condition d’arrêt d’entraînement est atteinte (voir Scénario d'entraînement) Eve est réinitialisée et repart pour 5 entraînements successifs avec 250 000 phases pour chacun d’eux. Si elle parvient à décrypter le message dans les 150 000 premières phases, alors l’entraînement est considéré comme un échec. Dans le cas contraire, il est considéré comme réussi.

Résultats

Le graphique ci-dessous montre une expérimentation réussie avec N=16 bits. Les deux réseaux de neurones commencent à 8 bits d’erreurs. Après un certains temps, Alice et Bob commencent à communiquer efficacement mais d’une façon qui permet à Eve de s’améliorer également. Vers la phase 10 000, Alice change sa stratégie pour contrer la progression d’Eve, expliquant ainsi pourquoi P_{Bob} remonte légèrement : il doit s’adapter à ce changement de stratégie. Vers la phase 15 000, les objectifs sont globalement atteints : P_{Bob} tend vers 0 et P_{Eve} vers 8. Les phases suivantes n'augmentent que légèrement le nombre d’erreurs d’Eve.

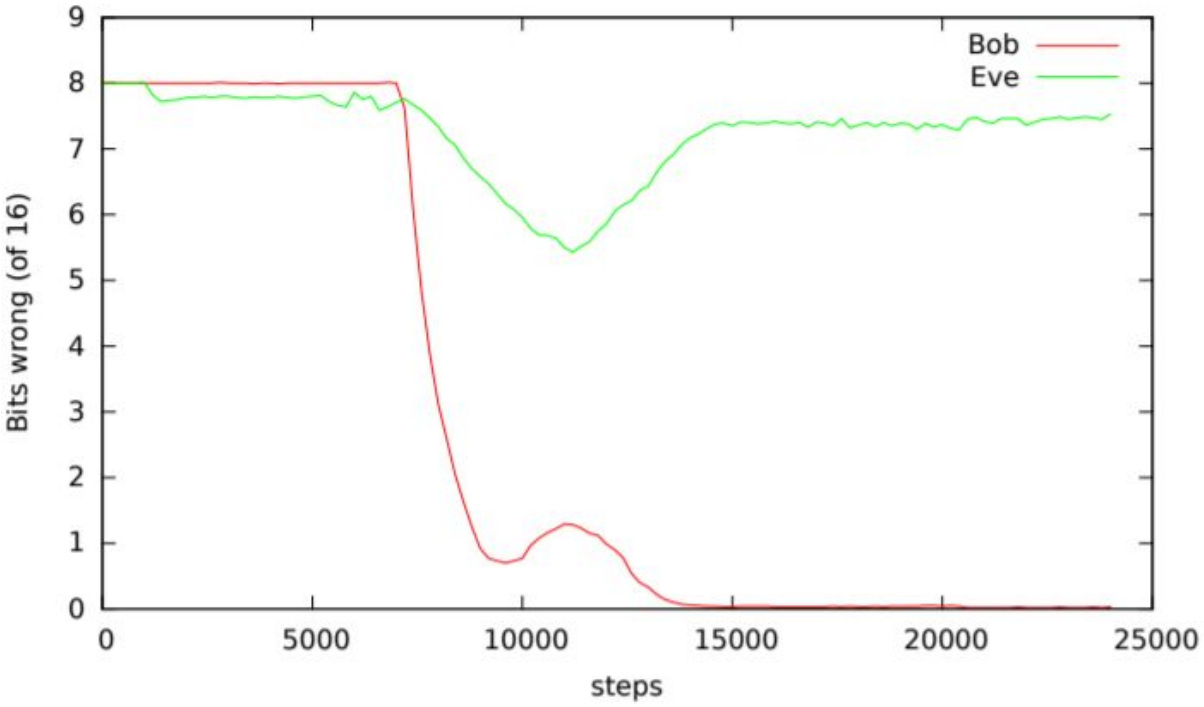

figure 4:résultat d'une expérimentation avec une clé de longueur 16 bits

Après plusieurs tentatives d'entraînement avec différents paramètres, et pour une longueur de N = 32 ou N = 64, l'entraînement réussit plus souvent avec une clé de 64 bits. Mais pour N = 16, 6 entraînements sur 12 ne réussissent pas : P_{Eve} < 7.3 bits d'erreur.

Les chercheurs ont voulu également observer quelles stratégies de chiffrement ont été utilisées. Ils se sont aperçus que le chiffrement était dépendant de la clé. En effet, la changer en gardant le même texte donne un chiffrement différent. Il dépend également du texte : le modifier en gardant la même clé donne, là encore, un chiffrement différent. La technique de chiffrement n'est pas un simple XOR. Premièrement, les valeurs sont des flottants compris entre 0 et 1. Deuxièmement, si 1 bit de la clé ou 1 bit du texte clair sont changés, les modifications se propagent dans le texte chiffré.

Conclusion

Dans cet article, les chercheurs ont démontré en quoi les réseaux de neurones peuvent apprendre à protéger leurs communications, sans prescrire de fonction cryptographique. Il a été également mis en évidence que la mise en compétition de plusieurs réseaux de neurones permet d'augmenter la robustesse du chiffrement.

Références

1. Abadi, M. and Andersen, D. (2016). *Learning to Protect Communications with Adversarial Neural Cryptography*. [online] arXiv.org. Available at: https://arxiv.org/abs/1610.06918 [Accessed 8 Nov. 2019].
2. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). *Generative Adversarial Networks*. [online] arXiv.org. Available at: https://arxiv.org/abs/1406.2661 [Accessed 8 Nov. 2019].
3. Medium. (2019). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. [online] Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [Accessed 8 Nov. 2019].