Elimination du biais dans les données utilisées pour le recrutement grâce à une approche adversariale

Jihane Bennis
jihane.bennis@student.ecp.fr

Rachel Guibert
rachel.guibert@student.ecp.fr

Abstract

Le Machine Learning est un outil pour construire des modèles de prédiction à partir de données d'entraînement recueillies auparavant. Quand des biais non désirés concernant des informations ethniques par exemple sont intrinsèques au dataset, des modèles bien entraînés peuvent perpétuer ces biais. Dans le cadre de l'inscription de cette problématiques dans le domaine du recrutement, nous présentons une chaîne de traitement permettant de représenter un CV ou une offre de manière débiaisée du genre. Le réseau est constitué d'un auto-encodeur qui prend en entrée X, un document textuel (CV) et fournit en sortie une représentation X_encoded sous une forme matricielle de taille 1024*, et d'un réseau adversarial qui essaie de prédire le genre de cette représentation (homme/femme). L'objectif est de minimiser la loss de l'auto-encodeur afin de préserver le maximum d'informations du CV, et de rapprocher l'accuracy du classificateur de genre à l'aléatoire (soit une valeur de* 0.5*) avec une architecture adversariale convergente. L'implémentation actuelle permet de trouver un système stable où l'accuracy du classificateur est aux alentours de* 0.6 *et la loss de la boucle adversariale est de* 0.25*. Le code est consultable sur GitHub [2]*

1. Introduction

L'utilisation montante d'algorithmes amenés à prendre des décisions avec de forts enjeux éthiques et impactant les activités humaines ont conduit à la création de nouvelles réflexions dans le domaine, centrées autour de l'éthique et de la justice. Parmi ces dernières, une des questions majeures est celle de la reproduction (voir l'augmentation dans le pire des cas) lors d'une prise de décision de biais et préjugés déjà présents dans les données [1].Au sein de ce travail, nous nous intéressons au domaine du recrutement pour lesquels des exemples récents nous ont montré les problèmes inhérents à la démarche. Dans le cadre de ce projet nous allons donc nous pencher sur la réduction du biais dans le domaine du recrutement, plus précisément des biais lié au genre dans des données textuelles (CVs) utilisés pour l'embauche d'interim. Ce projet s'inscrit dans un projet plus large de création d'un algorithme de mise en correspondance d'offres avec des candidats.

2. Etat de l'art

Il y a eu de nombreux travaux réalisés autour du débiaisement de divers types de données ou prédicteurs. *Débiaisement des word embeddings*[3] : à l'aide d'un réseau adversarial, les auteurs enlèvent le biais du genre dans un set de mots. En effet, ils ont remarqué que leur modèle associait des métiers particuliers (infirmière, ingénieur) selon le genre. Leur boucle adversariale a permis d'enlever ce lien. Ce papier a été une de nos grandes inspirations pour l'implémentation de notre architecture. *Entrainement adversarial* [5] : les auteurs ont été les précurseurs de l'utilisation de plusieurs réseaux de neurones avec des objectifs concurrents pour forcer l'un à contrarier l'autre en créant des images réalistes. Sur le même sujet, on peut aussi se référer à [7]. *Gradient à échelle multiple* [6] : Sorti au cours de notre projet (Mars 2019), ce papier présente des pistes d'amélioration de la convergence des réseaux adversariaux très intéressantes.

3. Approche

3.1. Chaîne du traitement de biais dans les CVs

Nous ne traitons pas ici directement la mise en correspondance des offres et des CVs, mais le débiaisement des données est intimement lié à la chaîne de traitement qui la suit. Notre recherche bibliographique nous a donc conduit à créer un algorithme de débiaisement lié à la chaîne de traitement suivante : deux réseaux de neurones profonds sont chargés d'extraire des représentations vectorielles des compétences et expériences (resp. des exigences) des candidatures (resp. des offres) (FIGURE 1). Ces deux représentations vectorielles sont ensuite mise en correspondance utilisant par exemple la technique des plus proches voisins.

Notre intervention se fera sur l'extraction de compétences du CV : l'idée sera donc d'obtenir des

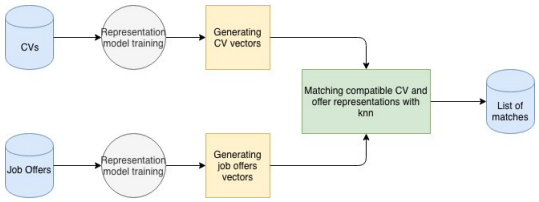


FIGURE 1: Schéma du processus de mise en correspondance étudié

représentations vectorielles *débiaisées* de ceux-ci.

Pour cela, nous avons d'adopter une approche s'inspirant des GAN (Generative Adversarial Network)[7] et des autoencodeurs. L'idée des GAN est d'entraîner en parallèle deux réseaux de neurones entraînés pour générer des images réalistes : une des applications les plus connues de cette technique est le site "This person does not exist", où sont exposés des visages générés de personnes qui n'existent pas. Dans un GAN, un premier réseau, le **générateur** génère des images à partir d'images d'entraînement et un second réseau, le **discriminateur** est un classifieur dont le but est de déterminer si l'image du générateur est réelle ou non. La classification est ensuite prise en compte pour les étapes suivantes d'entraînement du générateur. A terme, le générateur crée des images fausses, qui seront capables de tromper le classifieur.
Les autoencodeurs sont eux aussi des réseaux neuronaux basés sur deux sous réseaux : un premier réalise un **encodage**, qui permet de réduire la dimensionalité des entrées initiales, puis un second réalise le **décodage**, c'est à dire le retour à l'entrée initiale à partir de ce nouveau vecteur.

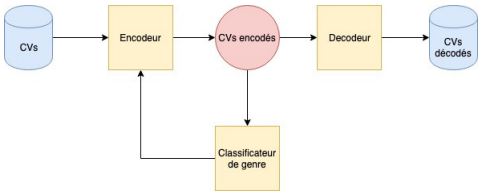


FIGURE 2: Schéma du processus d'élimination du biais

L'utilisation de ces deux types de réseaux (FIGURE 2) nous est conseillée par la littérature [4] et répond de la double exigence que nous rencontrons : nous cherchons à la fois à débiaiser l'encodage, mais à garder le plus d'informations possibles dans les CVs, ce que l'on peut contrôler grâce au décodage.

3.2. Implémentation de la chaîne de traitement

Pour la classification du genre, nous avons décidé d'implémenter un perceptron multicouches (MLP).Il s'agit d'un type de réseau neuronal formel organisé en plusieurs couches au sein desquelles une information circule de la couche d'entrée vers la couche de sortie uniquement. L'intérêt d'un MLP est que ses couches sont totalement connectées, ce qui permet de préserver toute l'information mais peut demander des ressources en calcul plus grandes. Étant donné que nous ne considérons que l'aspect textuel des CVs et la taille de notre dataset, la capacité de calcul a été suffisante. Les différentes composantes du MLP sont les suivantes :

- *Entrée :* La représentation tokénisée des CVs. D'un point de vue plus pratique, l'entrée est composée d'un vecteur de taille 1024, qui correspond à la longueur que nous avons fixée pour la représentation de chaque CV.
- *Couche Cachée :* Nous avons opté pour une couche d'activation ReLu, largement utilisée dans l'état de l'art.
- *Couche de sortie :* Nous avons décidé d'appliquer une normalisation *softmax* et avons fixé la forme de sortie à un vecteur de deux dimensions. La sortie correspond ainsi à un vecteur *(male, female)* contenant la prédiction du réseau neuronal sous la forme d'une probabilité pour chacune des deux classes.
- *La fonction de coût :* Nous avons également choisi une fonction de coût largement utilisée dans le domaine de la classification : la *cross entropy*

Afin de mesurer l'efficacité de notre classificateur, nous nous sommes intéressés à la métrique *accuracy*.

L'implémentation d'un auto-encodeur consiste en l'implémentation de deux réseaux neuronaux. Nous avons décidé d'implémenter deux MLP à une seule couche cachée, chacun avec des entrées/sorties correspondant à sa fonctionnalité. Ainsi, nous avons :

- *Réseau neuronal de l'encodeur :* ce MLP a pour entrée le CV tokénisé, soit un vecteur de dimension 1810 (*taille fixée lors du preprocessing*) et une sortie correspondant à la représentation matricielle des CVs, de taille 1024.
- *Réseau neuronal du décodeur :* ce MLP prend en entrée la représentation matricielle générée par l'encodeur, de taille 1024. Sa sortie a les dimension du CV tokénisé, soit une dimension de 1810.

Chacun de ces réseaux neuronaux a été implémenté avec une fonction d'activation ReLu, et une fonction de coût sous forme d' **erreur quadratique moyenne** ; un choix basé sur l'état de l'art des pratiques en apprentissage profond.

3.3. Implémentation de la boucle adversariale

Tout d'abord, nous avons ajouté **une fonction de coût caractéristique de la boucle adversariale**. Au sein de cette boucle, nous prenons en compte les fonctions de

coût de l'autoencodeur et celle du classificateur. Le but est de minimiser le coût de l'autoencodeur, pour s'assurer qu'il restitue le plus d'information possible, et en parallèle forcer la fonction de coût du classificateur à augmenter, ce qui nous permet de diminuer les performances du classificateur. En effet, rappelons-le, le but de la boucle adversariale est d'obtenir un autoencodeur le plus efficace possible, générant une représentation (*sortie de l'encodeur*) dont le genre est indiscernable - ce qui se traduit par une médiocre performance du classificateur de genre sur la représentation en cours de débiaisement. La fonction t adversariale adoptée est donc de la forme :

$$Loss_{adversarial} = Loss_{autoencoder} - \beta * Loss_{classifier}$$

où β est un paramètre, $Loss_{autoencoder}$ et $Loss_{classifier}$ les fonctions de coût de l'autoencodeur et du classificateur.

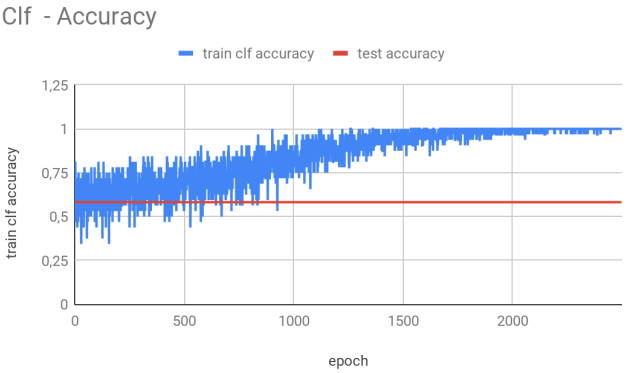
Ensuite, lors de l'entraînement du modèle, nous nous sommes assurés d'**alterner l'apprentissage** de l'autoencodeur et de celui du classificateur. Par exemple, on entraîne l'autoencodeur une fois sur deux, on en profite pour mettre à jour la fonction de coût adversariale, puis on entraîne le classificateur sur les nouvelles représentations générées par l'encodeur. *La fréquence d'alternance est un paramètre que l'on étudiera plus tard.*

4. Expériences

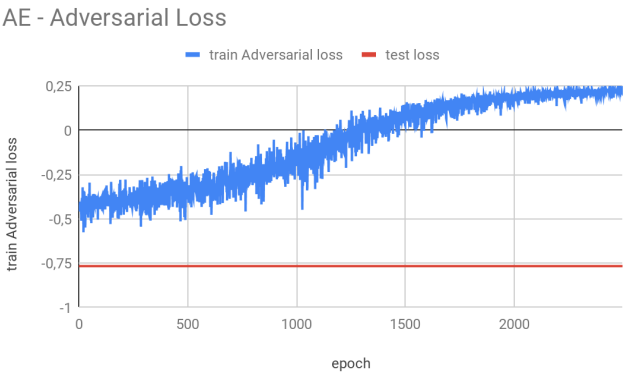
Nous utilisons pour entraîner et tester notre algorithme un dataset de 12000 CVs sous format textuel, labellisés par nos soins. Ca dataset n'est pas équilibré : environ $2/3$ des CVs sont masculins et $1/3$ féminins. Les CVs ont ensuite été tokenizés, et les vecteurs les représentants normalisés, pour réaliser les expériences.
L'entraînement sur les données se fait par *batch* afin d'éviter un sur-apprentissage de l'ensemble d'apprentissage.
Comme expliqué précédemment, l'entraînement des modèles se fait par alternance : l'autoencodeur puis le classificateur. Au fil de l'entraînement, nous affichons ainsi les *Loss* de l'autoencodeur et de la boucle adversariale, et l'Accuracy du classificateur pour bien suivre les performances de chaque architecture choisies. Nous avons en effet construit différents modèles en jouant sur différents hyperparamètres : *β, taille du batch, Pas d'apprentissage des modèles, Fréquence d'alternance de l'apprentissage des modèles* .Ces modèles ont rencontrés deux problèmes fréquents :

— Nous avions en général de gros problèmes d'**overfitting** sur l'entraînement de nos modèles, comme le montre la FIGURE 3a. En effet, on s'aperçoit d'abord l'accuracy du classificateur ne cesse d'augmenter pour atteindre 1 pendant l'entraînement, ce qui est contraire à notre besoin, mais aussi que cette valeur ne correspond pas du tout à la valeur de test, quasiment situé à l'aléatoire, ce qui est un bon signe d'overfitting.

— Dans la figure FIGURE 3b, on voit un cas où le "débiaisement" n'est pas efficace. En effet, la loss de notre modèle est croissante, alors qu'on la souhaiterait décroissante et si possible négative.



(a) Exemple d'overfitting



(b) Débiaisement non efficace

FIGURE 3: Les problématiques du modèle

4.1. Influence de la taille du batch

Les figures FIGURE 4 montrent bien que la taille du batch a une influence majeure sur l'overfitting : plus le batch est grand, plus le risque d'overfitting est important. Lorsque le batch est égal à 10, on a par contre une situation assez intéressante : la loss de notre modèle adversarial est quasi constante et proche de la valeur de test. On voit aussi dans ce cas que cette stabilité n'implique pas nécessairement une stabilité de l'accuracy : on est dans un système assez fortement oscillant, mais qui est centré sur la valeur de test, ce qui est assez intéressant pour nous. Le système a alors une

certaine **stabilité**.

4.2. Influence du pas d'apprentissage

Le deuxième paramètre dont nous avons étudié l'influence est le rapport des pas d'apprentissage du classifieur et de l'autoencodeur (FIGURE 5). $LR = \frac{Pas\ d'apprentissage\ de\ l'autoencodeur}{Pas\ d'apprentissage\ du\ classifieur}$. On remarque que lorsque ce rapport est grand, on a de l'overfitting, ce qui s'explique assez aisément par le fait que le classifieur avec un pas plus petit est beaucoup plus sensible et "apprend mieux". On choisit donc les pas d'apprentissage de telles manières à avoir $Pas\ d'apprentissage\ de\ l'autoencodeur \leq Pas\ d'apprentissage\ du\ classifieur$.

4.3. Influence du paramètre β

Nous avons aussi étudié l'importance du paramètre β sur le comportement de notre modèle. Pour celà, nous avons étudié l'évolution de la loss de la boucle adversariale d'une part, et de l'autoencodeur d'autres part. En effet, l'objectif est à la fois de diminuer la loss de la boucle adversariale, sans amener une trop grosse perte d'information, c'est à dire une loss de l'autoencodeur trop importante. Cet objectif nous permet rapidement d'éliminer le cas de la FIGURE 6c où on voit que la loss de la boucle adversariale ne fait qu'augmenter ce qui montre que l'accuracy de notre classifieur ne fait qu'augmenter elle aussi. La figure FIGURE 6a n'est pas non plus un cas idéal, car la loss de l'autoencodeur augmente beaucoup et n'atteint pas de stabilité. Le cas le plus idéal est celui où $\beta = 1$ (FIGURE 6b), car il nous montre que la loss du classifieur est constante et positive, ce qui donne une certaine stabilité au modèle et nous permet deprévoir ces performances.

4.4. Influence de l'alternance d'apprentissage

A partir des cas les plus idéaux que nous avions trouvé, c'est à dire de telle manière que le classifieur soit oscillant autour d'une manière stable et une loss de la boucle adversariale constante et proche de 0, nous avons décidé de jouer sur le nombre d'entrainement du classifieur et de l'autoencodeur, espérant diminuer notre loss et l'oscillation du classifieur. En diminuant le nombre d'entraînement du classifieur, nous avons de manière assez surprenante, une augmentation de la loss de la boucle adversariale et peu d'influence sur les oscillations (FIGURE 7). Pour le moment, cette piste n'est pas concluante et mériterait plus d'exploration.

4.5. Pistes d'améliorations des modèles

— Qu'il s'agisse de l'autoencodeur ou du classificateur de genre, nous avons utilisé des *Multi Layer Perceptron (MLP)*. Les MLP se distinguent par leur caractère polyvalent et facilement implémentable. Cependant malgré leur polyvalence, les MLP ne sont pas nécessairement les réseaux de neurones offrant les meilleures performances en matière de NLP.
Un type de réseau est particulièrement adapté aux problèmes textuels tels que celui que nous tentons de traiter. Il s'agit des réseaux *LTSM*. Les LSTM sont des réseaux appartenant à la catégorie des Recurrent Neuronal Networks (RNN). Ces derniers sont très adaptés lorsqu'il s'agit de traiter des données séquentielles comme un texte par exemple.

— Notre fonction de coût présente l'inconvénient qu'il est difficile d'estimer une valeur idéale pour le coût issu du classificateur. En effet on souhaite que le coût de l'autoencodeur soit proche de 0 et que le coût du classificateur soit élevé, cependant on ne sait pas quelle valeur serait assez élevée pour considérer le classificateur incapable de déceler le genre dans les CVs encodés.
Ce faisant, il serait intéressant d'essayer d'implémenter une boucle reposant sur la fonction de coût suivante :

$$Loss_{adversarial} = Loss_{autoencoder} + \alpha * Accuracy_{classifier}$$

Ici nous avons en tête une valeur idéale pour les deux composantes de notre fonction de coût. Idéalement la fonction de coût de l'autoencoder aurait une valeur proche de 0. Nous voudrions aussi que le classificateur ne puisse pas plus déceler le genre que le hasard, c'est-à-dire que l'Accuracy du classificateur soit proche de 50%. La fonction de coût adversariale tendrait donc idéalement vers $0.5 * \alpha$.

— Dans notre réseau actuel, l'encodeur, le décodeur et le classificateur ne comportent chacun qu'une seule couche d'activation. Il serait intéressant d'observer les résultats que pourrait avoir l'ajout de couches de neurones dans l'un des trois réseaux. L'ajout de couche est réputé pour améliorer les performances d'apprentissage mais aussi d'augmenter le risque de sur-apprentissage.

— Enfin nous avons vu que nos réseaux étaient très sujets à l'overfitting, ce phénomène qui consiste en de très bonnes performances d'apprentissage mais une très mauvaise capacité de généralisation (résultats de test bien en deçà de ceux observés pendant d'entraînement).

5. Conclusion

Finalement, ce projet était un véritable défi éthique et technique au coeur de l'actualité sur lequel nous avons beaucoup aimé travailler. L'utilisation d'une boucle adversariale pour débiaiser un CV est pour nous une

piste intéressante mais qui n'aboutit pas rapidement à des résultats satisfaisants. En effet, malgré le test de différentes combinaisons des paramètres de nos modèles, nous n'arrivons pas à concilier débiaisement du genre et restitution satisfaisante des informations d'un CV sans faire de sur-apprentissage sur l'ensemble d'apprentissage. Il nous est donc nécessaire d'explorer d'autres pistes, tel la profondeur des réseaux neuronaux, l'équilibrage des données en entrée ou encore l'utilisation d'autres modèles pour représenter le CV à l'entrée de nos modèles.
Nous ne prenons en effet pas compte pour le moment du nombre d'occurrence d'un mot ni de son contexte.

Au delà de tester différents paramètres pour trouver un meilleur modèle, nous pouvons également explorer d'autres pistes tout aussi intéressantes. Nous avons en effet pensé à utiliser un classificateur de compétences au lieu de l'autoencodeur. La boucle adversariale permettrait ainsi d'extraire des compétences d'un CV tout en débiaisant le rapport entre les compétences extraites et le genre du CV.

Références

[1] A. N. Aylin Caliskan, Joanna J. Bryson. Semantics derived automatically from languagecorpora contain human-like biases. 2017.

[2] R. G. J. B. Unbiased Recruiting. `https://github.com/unbiased-recruiting/unbiased-recruiting`, 2019.

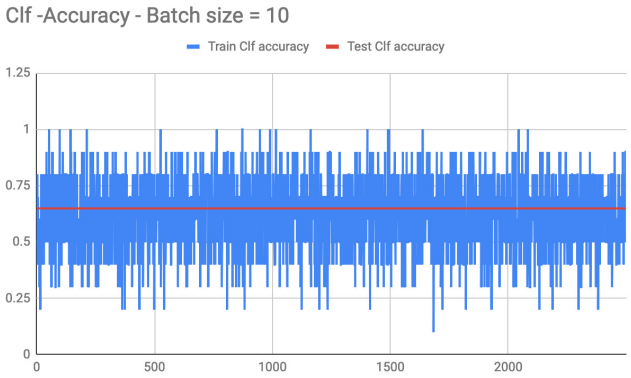
[3] K.-W. Z. J. Y. S. V. Bolukbasi, T.; Chang and A. T. Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. 2016.

[4] M. M. Brian Hu Zhang, Blake Lemoine. Mitigating unwanted biases with adversarial learning. 2018.

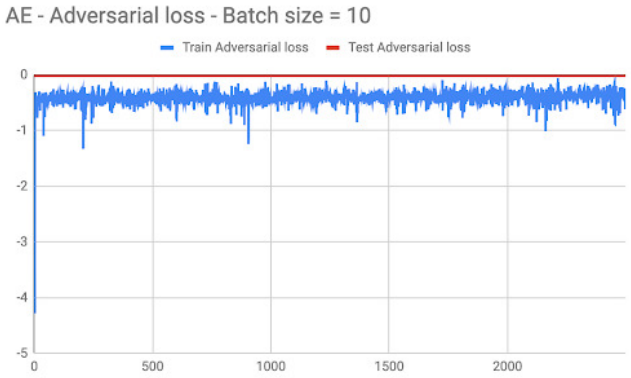
[5] J. M. M. X. B. W.-F. D. O. S. C. A. Goodfellow, I.; Pouget-Abadie and Y. Bengio. Generative adversarial nets. 2014.

[6] A. K. O. W. R. S. Iyengar. Msg-gan : Multi-scale gradient gan for stable image synthesiss. 2019.

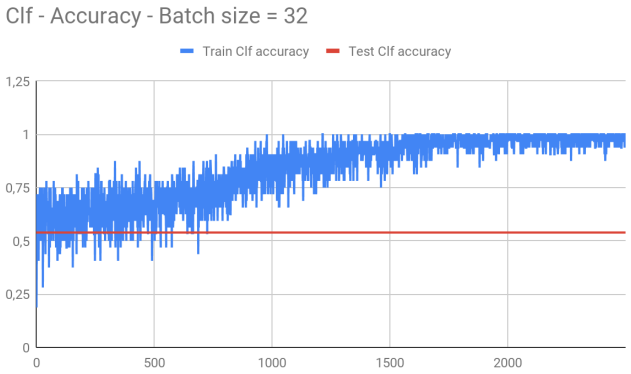
[7] Shrivastava. Learning from simulated and unsupervised images through adversarial training. 2017.



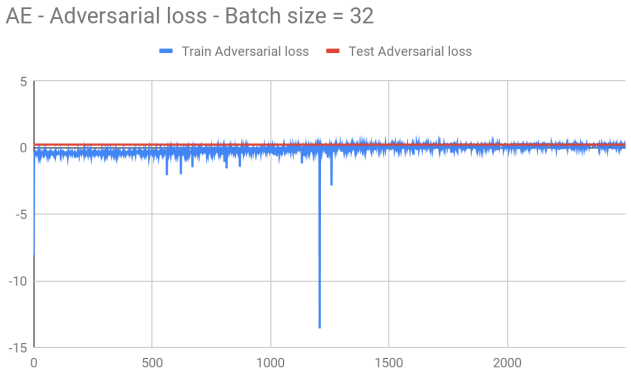
(a) Taille du batch de 10 : Accuracy du classifieur



(b) Taille du batch de 10 : Loss de l'autoencodeur

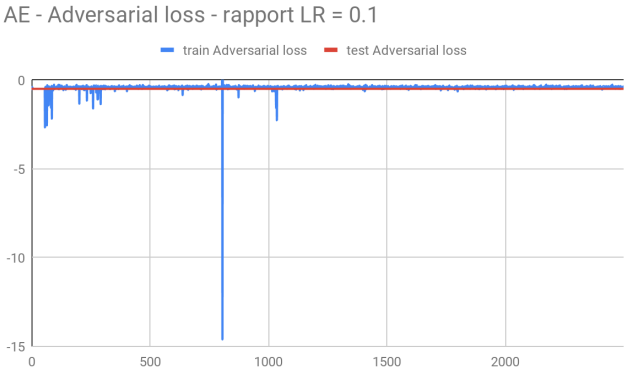


(c) Taille du batch 32 : Accuracy du classifieur

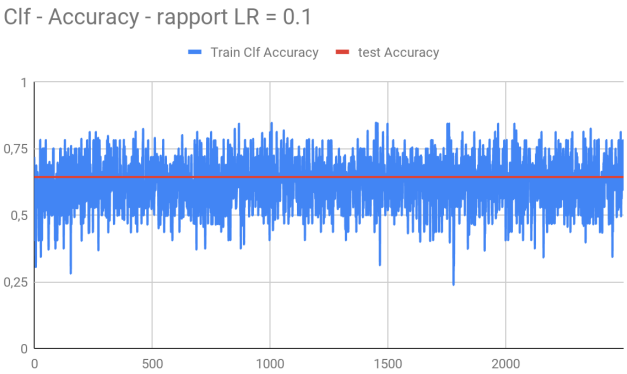


(d) Taille du batch 32 : Loss de l'autoencodeur

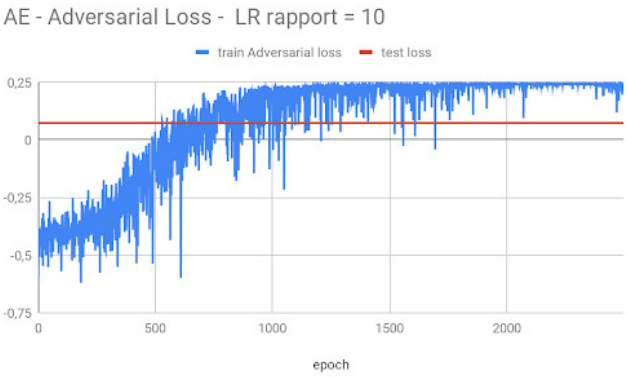
FIGURE 4: Influences de la taille du batch



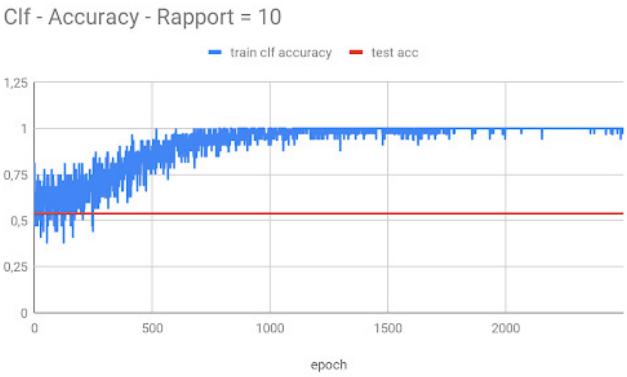
(a) Ratio des pas d'apprentissage de 0,1 : Loss de l'autoencodeur



(b) Ratio des pas d'apprentissage de 0,1 : Accuracy du classifieur

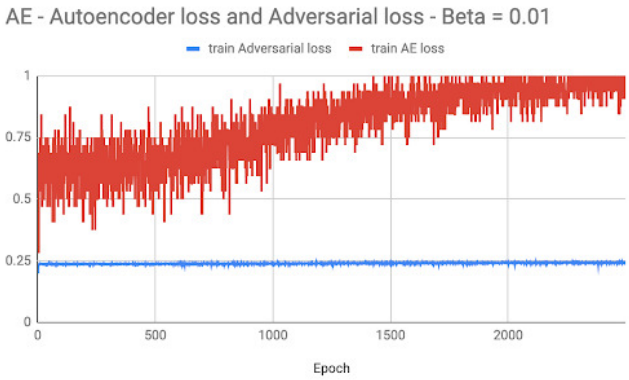


(c) Ratio des pas d'apprentissage de 10 : Loss de l'autoencodeur

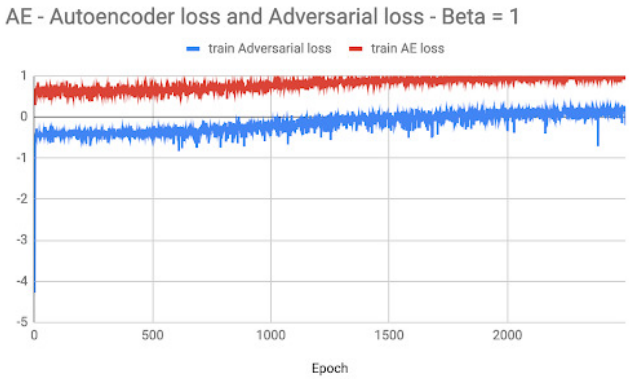


(d) Ratio des pas d'apprentissage de 10 : Accuracy du classifieur

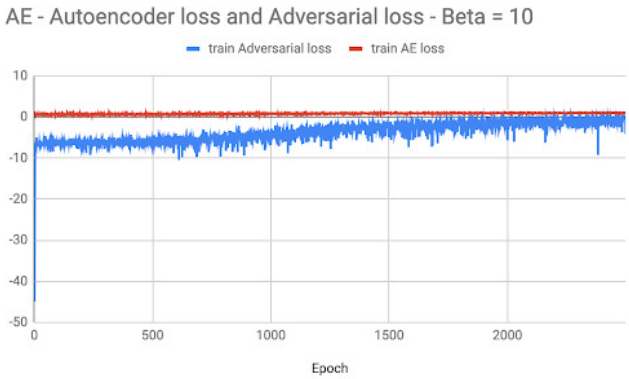
FIGURE 5: Influence des ratios de pas d'apprentissage



(a) β de 0,01

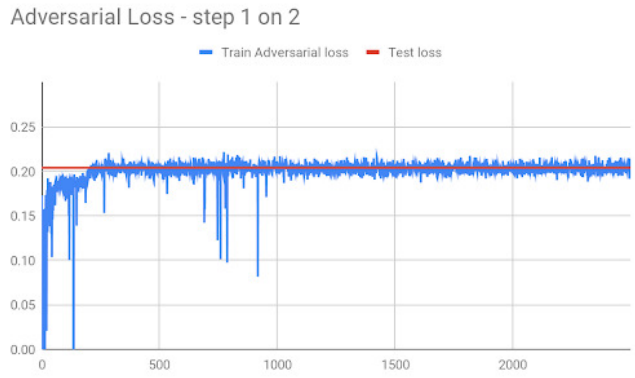


(b) β de 1

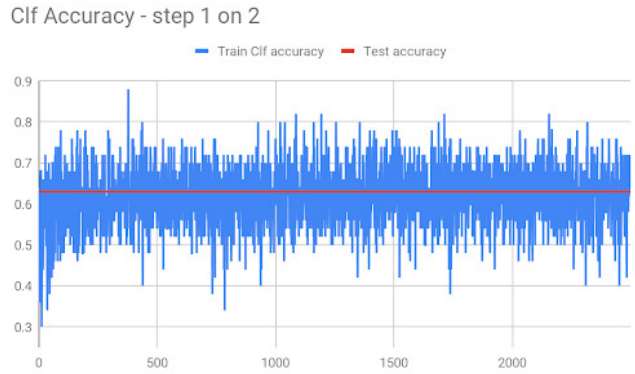


(c) β de 10

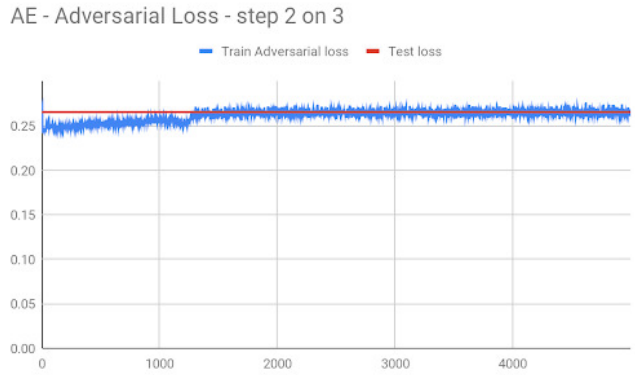
FIGURE 6: Influence de β



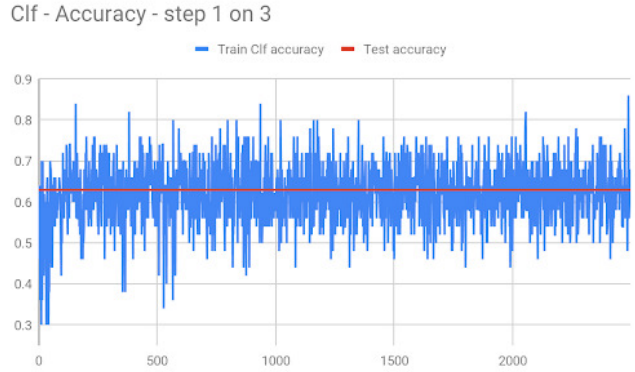
(a) Passage de l'autoencodeur 1 fois sur 2 : Loss de l'autoencodeur



(b) Passage de l'autoencodeur 1 fois sur 2 : Accuracy du classifieur



(c) Passage de l'autoencodeur 2 fois sur 3 : Loss de l'autoencodeur



(d) Passage de l'autoencodeur 1 fois sur 3 : Accuracy du classifieur

FIGURE 7: Influence de l'alternance d'apprentissage