Rapport de Projet
Modélisation en Neurosciences et ailleurs
Synaptic Sampling : A Bayesian Approach to Neural Network Plasticity and Rewiring

Louis Fournier

31 mars 2020

1 Introduction

L'article Synaptic Sampling : A Bayesian Approach to Neural Network Plasticity and Rewiring [1] a été publié à la conférence NIPS en 2015. Il est principalement une adaptation d'un papier plus long et détaillé publié dans PLOS en 2015, Network Plasticity as Bayesian Inference [2]. Il apporte un cadre mathématique nouveau à la plasticité synaptique en la considérant comme un problème bayésien, cherchant à sampler les paramètres dans une distribution postérieure au lieu d'une simple maximisation de vraisemblance. Ce cadre mathématique permet de bien justifier les recâblages constants observés dans le cerveau et leur variété, et donne des bons résultats pour les réseaux de neurones (notamment les Spiking Neural Networks), robustes à des modifications importantes.
Dans ce rapport, nous allons présenter ces articles en profondeur, et proposer une réimplémentation de Synaptic Sampling pour une machine de Boltzmann neuronale et analyser ses capacités d'apprentissages.

2 Théorie du Synaptic Sampling

2.1 L'apprentissage réseau comme problème bayésien

L'objectif d'apprentissage d'un réseau (génératif) est généralement la maximisation d'une distribution de vraisemblance $p_{\mathcal{N}}(x|\theta)$, qui représente la probabilité d'inputs x selon les paramètres θ du modèle. Il est alors relativement aisé de trouver des maximums locaux de $\theta^* = argmax_\theta p_{\mathcal{N}}(x|\theta)$ par des méthodes comme la descente de gradient.

Cependant optimiser seulement cette vraisemblance donne des résultats peu robustes, overfittant facilement, et peu naturels. Il serait donc meilleur d'obtenir des résultats plus robustes en apprenant directement la distribution postérieure $p^*(\theta)$, en prenant en compte la distribution des prior des paramètres du réseau, notée $p_{\mathcal{S}}(\theta)$. Le nouveau but d'apprentissage du réseau devient alors de déterminer :

$$p^*(\theta|x) \propto p_{\mathcal{S}}(\theta)p_{\mathcal{N}}(x|\theta)$$

Ce cadre bayésien d'apprentissage présente plusieurs avantages comme il sera montré plus tard. Les paramètres θ du réseau vont donc fluctuer constamment selon ces distributions de probabilités $p^*(\theta)$, restant localement dans une variété locale. Le réseau va donc "échantillonner" (sampling) ses paramètres selon cette distribution, d'où le nom de Synaptic Sampling de cette

théorie. Il est un bon cadre d'explication des mécanismes de plasticité neuronaux (synaptiques) comme nous allons le voir.

2.2 Une dynamique stochastique pour l'apprentissage

Le papier propose alors une dynamique stochastique des paramètres du réseau qui permet d'apprendre la distribution du postérieur. On suppose donc que tout les paramètres θ_i obéissent à la dynamique :

$$d\theta_i = \left(b(\theta_i) \frac{\partial}{\partial \theta_i} \log p_S(\theta) + b(\theta_i) \frac{\partial}{\partial \theta_i} \log p_N(x|\theta) + T b'(\theta_i) \right) dt + \sqrt{2Tb(\theta_i)} d\mathcal{W}_i$$

Avec $\mathcal{W}_i$ suivant un processus de Wiener (i.e. $\mathcal{W}_i^t - \mathcal{W}_i^s \sim Normal(0, t-s)$), T>0 la température et b>0 une fonction paramètre de vitesse de sampling.
Les paramètres θ_i évoluent donc selon trois termes. La distribution du prior, qui force des règles structurelles ; la distribution de la vraisemblance, qui vérifie la cohérence avec les inputs selon l'activité ; et un terme stochastique de mouvement brownien qui synthétise les différents facteurs stochastiques dans le développement des synapses.

Alors avec b strictement positive et p et $b \in C^2$ cette dynamique converge vers la distribution postérieure suivante :

$$p^*(\theta) \propto p^*(\theta|x)^{\frac{1}{T}}$$

qui est l'unique invariante par rapport au temps. La démonstration utilise le fait que cette équation est une équation de Fokker-Planck avec un terme de "drift" (le postérieur) et un terme de diffusion (stochastique).
Ce résultat est fort, et prouve qu'il est possible d'apprendre la distribution voulue avec un processus stochastique. Le facteur T de température qu'il reste dans la distribution trouvée $p^*(\theta|x)^{\frac{1}{T}}$ permet de gérer l'influence stochastique. Avec $T \to 0$, seuls les maximums de $p^*(\theta)$ restent, gardant un processus deterministe (Maximum A Posteriori) ; et si T est grand la distribution devient plus homogène. Pour $T = 1$ on retrouve la distribution voulue.

2.3 Le synaptics sampling séquentiel

Dans un cadre plus pratique, le réseau apprend d'une manière séquentielle et non avec la séquence d'inputs entière comme précédemment. En considérant que l'on présente les N inputs x^i chacun pendant τ_x unité de temps, on peut prouver qu'avec des vitesses de sampling petites ($\frac{1}{N\tau_x} \ll b(\theta_i)$), on trouve la même équation de changements de paramètres qu'avec la règle dynamique de batch :

$$\theta_i^{N\tau_x} - \theta_i^0 \approx N\tau_x \left(b(\theta_i^0) \frac{\partial}{\partial \theta_i} \log p_S(\theta^0) + b(\theta_i^0) \sum_{n=1}^{N} \frac{\partial}{\partial \theta_i} \log p_N(x^n|\theta^0) + T b'(\theta_i^0) \right) + \sqrt{2Tb(\theta_i^0)} (\mathcal{W}_i^{N\tau_x} - \mathcal{W}_i^0)$$

On voit bien la même règle que pour un apprentissage classique de descente de gradient de Maximum de Vraisemblance, avec l'ajout d'un terme de prior et un terme stochastique.

3 Applications

3.1 Restricted Boltzmann Machine

Une machine de Boltzmann restreinte est un modèle de réseau de neurones très simple. Elle est composée de deux couches : une de neurones visuels et une de neurones cachées. L'output

d'un neurone x_i est binaire, proportionnel à la probabilité $\sigma(\sum_j w_{ij}y_j + b_i)$ avec σ la fonction sigmoïde, b_i le biais du neurone et w_{ij} le poids synaptique.
Il est alors facile en utilisant l'approximation du gradient de la vraisemblance d'appliquer notre formule d'apprentissage des paramètres pour en faire un réseau utilisant le synaptic sampling. Les auteurs implémente ce réseau (dans le papier secondaire) dans le cadre de l'apprentissage d'images MNIST (chiffres manuscrits), et démontrent que l'utilisation d'un prior (bi-gaussien) améliore grandement la capacité de généralisation de ce modèle.
Nous avons réimplémenté ce réseau simple pour réaliser diverses expériences inspirées par ce papier que nous présentons dans la section 4.

3.2 Modèlisation de la motilité synaptique pour le synaptic sampling

Pour appliquer le synaptic sampling à des modèles neuronaux biologiques et à impulsions, on pose un modèle simple de motilité synaptique (le cadre du sampling permet des modèles plus complexes). On ne considère que des synapses excitatrices, avec une seule épine dendritique entre deux neurones maximales, définies par un seul paramètre θ_i, et la taille de l'épine dendritique représentée par ce paramètre est proportionelle à la quantité de changement synaptique. On considère donc w_i l'efficacité synaptique fonction de ce paramètre selon une exponentielle : $w_i = \exp(\theta_i - \theta_0)$.

Le paramètre θ_i représente donc la structure synaptique et son poids en même temps. Le prior considéré pour ce paramètre est gaussien. On obtient que un θ_i important par rapport à θ_0 implique que seul le terme d'activité aura un impact sur le paramètre ; et si θ_i est négatif ou faible par rapport à θ_0, seul les effets de prior et stochastiques influe fortement la dynamique, revenant à un processus de Ornstein-Uhlenbeck. On remarque donc également que le processus stochastique va principalement permettre la création de liens synaptiques mais très peu de déconnexions. Tous ces résultats sont cohérents avec les observations sur la motilité synaptique.

3.3 Implémentation dans un Spiking Neural Network

Le modèle proposé précédemment est ici appliqué à un modèle de réseau neuronal à impulsion, de style "winner-take-all" (WTA), permettant ainsi d'avoir une bonne approximation du gradient de la vraisemblance du réseau. Le potentiel de la membrane d'un neurone k est donné par $u_k(t) = \sum_i w_{ki}x_i(t) + \beta_k(t)$. Avec $x_i(t)$ l'input du neurone i, égal à la somme des réponses d'un kernel (double exponentiel) entre le temps t et ceux des différentes impulsions reçues $t_i^{(l)}$. Et $\beta_k(t)$ un courant d'adaptation homéostatique décroissant lentement, à la même structure que $x_i(t)$ mais avec des temps d'adaptations beaucoup plus longs. Alors la cadence de tir instantanée du neurone k sera $\rho_k(t) \propto \rho_{net} \exp(u_k(t))$, à normalisation près.

Ce réseau définit un modèle génératif implicite. Il est aisé d'approximer le gradient de la vraisemblance $\frac{\partial}{\partial w_{ki}} \log p_{\mathcal{N}}(x^n|w)$, et avec le modèle précédant, on peut alors écrire entièrement la règle de synaptic sampling séquentiel. On prend $T = 1$, et b constant $\ll 1$, et $S_k(t)$ représente le spike train du neurone k en t. On rappelle que le prior est considéré gaussien. Alors on a :

$$d\theta_i = b\left(\frac{1}{\sigma^2}(\mu - \theta_i) + Nw_i(S_k(t)(x_i(t) - \alpha\exp(w_{ki}))\right)dt + \sqrt{2b}d\mathcal{W}_i$$

3.4 Expériences démontrant la plasticité synaptique

L'expérience va cette fois être la création d'un réseau à deux couches, une d'inputs, séparé en deux prenants en inputs soit les images MNIST précédentes, soit des cochléogrammes de

fichiers audio de personnes lisant les chiffres. On associe 4 réseaux WTA cachés pour chacun de ces inputs. Les neurones à l'intérieur de ces réseaux sont connectés, ainsi qu'avec les neurones du réseau WTA équivalent pour l'autre input, et bien sûr les inputs associés. Des connexions à d'autre neurones sont autorisées. Ce réseau va donc être entraîné de manière non supervisée à associer un cochléogramme de "1" ou de "2" avec une représentation visuelle du chiffre, et vice-versa. Il est possible d'inférer une reconstruction visuelle du réseau à partir d'un fichier audio.

Le réseau présente de très bonnes capacités d'apprentissage, mais c'est sa robustesse et sa capacité d'adaptation qui est soulignée par l'article. Ils vont tenter des lésions du réseau obtenu, tout d'abord en supprimant des neurones qui s'étaient spécialisées dans la classification de "2", puis plus tard en retirant toutes les synapses présentes jusque là. Dans les deux cas, le réseau est capable de s'adapter et recréer des liaisons fonctionelles dans le réseau très réduit, permettant toujours une bonne reconstruction. La vitesse à laquelle le réseau peut s'adapter et se réparer est considérablement réduite pour un réseau déterministe (sans le terme stochastique). Il est intéressant d'observer avec une PCA sur un ensemble de 35 paramètres que les paramètres semblent samplé sur des variétés à basse dimension, qui vont complètement changer après chaque liaison.

Une expérience supplémentaire est effectuée dans le papier originel, reprenant le principe de l'expérience de la machine de Boltzmann : on a cette fois ci seulement la partie visuelle du réseau. Dans cette expérience l'apprentissage se fait en trois phases. Le dataset est d'abord composé uniquement de 1, puis de 1 et de 2, et enfin uniquement de 1 à nouveau. On peut alors voir l'évolution des synapses redevenant actives à l'apparition de 2 par exemple, devenant bien plus fortes quand leur évolution était jusque là influencé uniquement par le terme de prior et stochastique.

4 Implémentation personnelle de Synaptic Sampling pour une MBR

Dans cette partie, nous allons maintenant présenter notre implémentation de l'expérience proposée par l'article de Synaptic Sampling pour une Machine de Boltzmann Restreinte (MBR). La majorité des valeurs des paramètres prises sont celles proposées dans l'annexe de l'article original.

4.1 Préparation du Dataset

Le dataset MNIST est téléchargé depuis Tensorflow [3], et le dataset "MNIST audio" est tiré du github [4]. Les images sont prêtes à être utilisés après normalisation, étant alors sous la bonne forme. Pour l'audio cependant, on utilise la librairie Spectral [5] pour extraire les Mel-Frequency Cepstral Coefficients des fichiers audio (77 filtres, plus de précisions dans le code), les résultats sont en suite "padded" pour avoir une taille homogène, puis redécoupés et normalisés pour avoir des inputs acceptables. Les images sont séparés par label de chiffre, puis séparés de manière aléatoire en datasets d'entraînement et validation (80/10).
Il faut cependant binariser ces inputs pour le réseau. Cela peut être fait pour les images en considérant la valeur d'un pixel (normalisé) comme sa probabilité d'être 1, et ainsi ressortir une image binaire de l'image initiale. Cependant cela est beaucoup plus difficile pour le MFCC audio, ses valeurs étant beaucoup plus délicates à binariser. Nous avons essayé de choisir un threshold de 0.87 aux valeurs du MFCC pour obtenir un pixel valant 1, sans valeur aléatoire.

4.2 Détails d'implémentation

Pour pouvoir effectuer un entraînement, nous avons toujours besoin de savoir la valeur du gradient de la vraisemblance. Pour cela, nous allons utiliser l'algorithme de "contrastive divergence" présent dans [6]. Après un passage d'un input , on sort la valeur des neurones cachées z_i, puis on ressort à partir de ces valeurs la valeur des neurones visuelles x_j. On va continuer cette boucle 5 fois de plus pour en ressortir les outputs de "reconstructions" des neurones, $\hat{x}_j$ et $\hat{z}_i$. On peut alors calculer une valeur approximée des gradients de vraisemblances : $\frac{\partial}{\partial b_i^{hid}} \log p_{\mathcal{N}}(x^n|\theta) \approx z_i^n - \hat{z}_i^n$ et $\frac{\partial}{\partial w_{ij}} \log p_{\mathcal{N}}(x^n|\theta) \approx z_i^n x_j^n - \hat{z}_i^n \hat{x}_j^n$. En reprenant les formules des parties précédentes, on obtient alors avec $\eta = b\Delta t$ et ν_i^t un bruit Gaussien centré normé :

$$\Delta b_i^{hid} = \eta N(z_i^n - \hat{z}_i^n) + \sqrt{2\eta}\nu_i^t$$

$$\Delta b_j^{vis} = \eta N(x_j^n - \hat{x}_j^n) + \sqrt{2\eta}\nu_j^t$$

$$\Delta w_{ij} = \eta \left(\frac{\partial}{\partial w_{ij}} \log p_{\mathcal{S}}(w) + N(z_i^n x_j^n - \hat{z}_i^n \hat{x}_j^n) \right) + \sqrt{2\eta}\nu_{ij}^t$$

On a $\eta = 10^{-4}$ et $N = 100$. On initialise les paramètres : les biais visuels (784), les biais cachés (9) et les poids synaptiques (784*9) (et 616 au lieu de 784 pour l'audio). Les valeurs sont prises aléatoirement selon une distribution gaussienne de variance 0.25 et de moyenne 0 pour le poids et -1 pour les biais.
Comme pour l'expérience de l'article, il faut ensuite choisir un prior. On choisit de la même manière soit un prior uniforme (de gradient nul) soit une mixture de 2 gaussiennes de moyennes 1 et 0 et de variance de 0.15. Nous calculons la valeur du gradient du logarithme de ce dernier prior (indépendant entre les synapses) :

$$\frac{\partial}{\partial w_{ij}} \log p_{\mathcal{S}}(w_{ij}) = \frac{\frac{x-\mu_1}{\sigma_1} \exp(-\frac{(x-\mu_1)^2}{2\sigma_1^2}) + \frac{x-\mu_2}{\sigma_2} \exp(-\frac{(x-\mu_2)^2}{2\sigma_2^2})}{\exp(-\frac{(x-\mu_1)^2}{2\sigma_1^2}) + \exp(-\frac{(x-\mu_2)^2}{2\sigma_2^2})}$$

Il suffit donc maintenant d'implémenter la sortie des neurones visuelles et des neurones cachées, qui revient à calculer la valeur de la sigmoïde présentée dans les parties précédentes, et comme précédemment à considérer qu'il s'agit d'une probabilité d'obtenir la valeur 1 pour l'output. Une étape d'entraînement devient alors : fournir un input au réseau ; en sortir l'output caché, puis commencer 5 fois la boucle de propagation des sorties. On ressort $x_j, \hat{x}_j, z_i, \hat{z}_i$. On met ensuite à jour les paramètres du réseau selon les règles définies ci-dessus. Pour évaluer l'erreur de la reconstruction du réseau, nous calculons la moyenne de la valeur absolue de la différence de l'input et de la première reconstruction visuelle (cela correspond à la L1 loss).

4.3 Résultats préliminaires sur MNIST

On peut alors entraîner le réseau et observer ses capacités de reconstructions. L'entraînement est fait de manière séquentielle sans batch. Les tailles des datasets varient selon les labels, mais sont environ de taille 6000. La vitesse d'entraînement dépend beaucoup des paramètres. Pour des images et 9 neurones cachées, le réseau entraîne environ 5 ou 6 inputs par seconde. L'entraînement total pour un prend une quinzaine de minutes.

Nous montrons dans la Figure 1a les courbes d'erreur d'entraînement et de test (calculée tous les 500 inputs) pour les 10 chiffres différents. On peut observer que les résultats sont assez

variables pour les différents labels. Cependant tous les entraînements démontrent de très bons résultats avec une bonne généralisation (pas d'overfitting sur les données de validation). Nous présentons également 2 exemples de reconstruction pour chaque chiffre dans la Figure 1b. On peut ainsi observer grâce aux courbes d'erreurs et aux exemples que le réseau a le plus de difficultés avec les chiffres comme 0, 2, 6 et 8, les courbes très variées du dataset semblant être une source d'erreur constante. En revanche, les chiffres 1 et 7 donnent les meilleurs résultats, probablement car leur disposition est très souvent la même ou très légèrement déplacée dans les différentes images. Les résultats sont donc cohérents.

4.4 Apprentissage de données audio

Nous pouvons voir dans la Figure 2 nos résultats sur l'apprentissage de données audio. Comme prévu, la binarisation des données MFCC est très difficile et donne des résultats difficile à apprendre. Il est donc peu surprenant que la généralisation du modèle est très mauvaise comme on peut le voir avec le score de validation, et les reconstructions approximatives du réseau. Il serait peut être possible d'adapter ce modèle de manière à ce qu'il puisse lire des données non binarisées.

4.5 Limites du nombre de neurones et tentative de deuxième couche

Pour tester les limites d'apprentissages de ce modèle, nous avons tenté de réduire le nombre de neurones cachées de 9 à 3, et enfin à 1. Les résultats des erreurs sont montrées sur la Figure 3. Il est alors suprenant d'observer que la loss d'entraînement est très similaire pour les 3 réseaux, mais que la loss de validation montre bien que l'apprentissage est beaucoup moins général. On présente également un exemple de résultat avec 1 neurone et le même résultat pour 9 neurones, montrant le manque de robustesse du réseau avec seulement un neurone caché.

Par curiosité, nous avons voulu tenter une implémentation d'une deuxième couche de neurones cachées au réseau, plus profonde. Pour cela, nous avons simplement rajouté une liste de biais pour cette deuxième couche et une liste de poids synaptiques entre les neurones cachées de la couche 1 et 2. Pour propager le réseau dans le modèle, il suffit alors de descendre à la deuxième couche cachée, puis remonter à la première puis à la couche visuelle, d'une manière analogue au modèle initial. De la même manière, les poids sont mis à jour d'une manière analogue avec les mêmes règles que précédemment en utilisant les valeurs de la première et deuxième couche cachée pour trouver l'approximation du gradient de la vraisemblance. On ne modifie pas les autres règles d'apprentissages des couches visuelle et cachée n°1. On présente quelques outputq de ce modèle avec 9 neurones cachées dans la première couche et 5 dans la deuxième dans la Figure 4a, et les courbes d'erreur dans la Figure 4b. Les résultats sont clairement moins bon que précédemment, avec un surplus d'erreur. Il semble que l'analogie ne soit pas suffisant pour créer cette deuxième couche et que la règle d'apprentissage n'est maintenant plus valable. Il reste intéressant de vouloir étudier l'impact du Synaptic Sampling sur un réseau plus profond.

4.6 Évolution des paramètres

Pour apprendre plus en détail sur l'entraînement du réseau au fil du temps, nous avons décidé d'observer l'évolution de certaines neurones visuelles. Ces courbes sont visibles sur la Figure 5a Nous avons pris les valeurs 0,0, 14,14 et 20,10 à observer car ils représentent des pixels très différents. Le pixel 0,0 est presque toujours nul et on devrait donc voir les neurones représentant ce pixel avec des valeurs très basses. A l'opposé le pixel 14,14 qui représente le milieu de l'image est presque toujours allumé sur les images. Enfin le pixel 20,10 est un cas plus

particulier car présent sur un bon nombre d'images mais dépendant de l'orientation du 1, il est donc plus complexe. Les résultats sont un peu moins clairs. Pour 0,0, la distribution des poids est assez uniforme mais le plus important et le biais, dont la valeur est extrèmement basse à -5 à la fin de l'entraînement. Cela assure une annulation presque totale. Pour 14,14, le biais est surprenamment bas, après une baisse soudaine. Cependant les poids synaptiques sont beaucoup plus fort et compensent ce biais, donnant des résultats cohérents. Enfin pour 20,10, les poids et biais sont assez difficiles à interpréter et neutres, montrant bien la complexité de sa position.

Nous avons également choisi de compter à chaque étape le nombre de neurones dont le biais est supérieur à -1. Nous voulions ainsi recréer l'expérience du papier qui était de compter le nombre de synapses actives dans les Spiking Neural Networks. Il est plus difficile de le compter ici dans une situation où les biais sont un autre paramètre à prendre en compte dans les calculs. Nous avons donc opter pour observer ce nombre simple et son évolution. Il est représenté pour l'entraînement sur le dataset de 1 sur la Figure 5b. On remarque alors une distribution similaire que dans l'expérience du papier : le nombre de neurones avec un biais supérieur à -1 tombe très rapidement. Il semble ensuite remonter lentement. Cela pourrait signifier que l'on a une grande quantité de pixels inactifs ou plus difficiles à activer, ce qui est assez cohérent avec les inputs. Cependant on ne peut pas conclure, car cela ne prend pas en compte les poids synaptiques.

4.7 Différence entre les priors

Nous avons testé différentes distribution de priors pour comparer les résultats. Les courbes d'erreur sont visibles sur la Figure 6, ainsi que la distribution en histogramme des valeurs des poids synaptiques. On remarque une distribution assez bien respectée pour le prior bi gaussien, avec cependant une distribution bien différente que celle proposée dans le papier ce qui est surprenant (Beaucoup plus de valeurs négatives que proches de 0). On remarque que la distribution des poids synaptiques sans prior est une gaussienne, comme sur l'exemple du papier. Enfin nous avons essayé d'implémenter un prior tri gaussien, avec une troisième gaussienne entre les deux première. Cependant elle semble avoir été trop proche, et le prior n'est pas très apparent. Dans les trois cas, il est également surprenant de voir des résultats très similaires, ce qui n'est pas ce à quoi on devrait s'attendre.

4.8 Robustesse au changement de dataset

Inspirés par une des expériences du papier, nous avons voulu observer les capacités d'adaptation du modèle si on lui changeait les datasets. Nous avons donc créé un entraînement en trois phases. Nous passons tout d'abord en entrée le dataset de 1, puis le dataset de 2, et à nouveau le dataset de 1 entièrement. Les courbes d'erreur sont visibles en Figure 7a. Comme il était prévisible, la loss monte soudainement pour le dataset de 2, mais le réseau semble apprendre sur le nouveau dataset. L'apprentissage est cependant beaucoup plus lent que avec un réseau initial. Quand on revient sur le dataset de 1, on voit que le réseau a très peu perdu en score en apprenant à distinguer les 2, et il poursuit son entraînement facilement.
On peut également observer les poids et biais pour deux pixels, en 14,14 et 20,10, sur la Figure 7b. Il est intéréssant de voir une quasi stagnation (stochasticité exclue) en 14,14 pendant le dataset de 2, mais un assez grand changement du biais et de certains poids en 20,10, un pixel qui était plus ambigu pour les 1 mais souvent plus actifs pour les 2. Enfin, on peut également observer le résultat final du réseau sur 2 exemples, un 1 et un 2, dans la Figure 7c. On peut alors être surpris de voir que les résultats sont assez similaires, et que le réseau a bien remémoré la présence d'un 2 précédemment. Les résultats du 1 sont cependant légèrement moins bon que normalement, car des neurones doivent toujours être conservées pour les 2.

4.9 Robustesse aux lésions de paramètres

De la même manière, nous avons voulu créer une expérience relativement similaire à celle effectuée dans le papier originel sur la robustesse du réseau aux lésions. Pour cela, à deux étapes de l'apprentissage (2000 et 4000), nous réinitialisions de manière aléatoire la moitié des poids synaptiques et biais neuronaux. On observe le résultats sur les courbes d'erreur dans la Figure 8a. On observe des pics très forts sur l'erreur de validation, mais des changements moins impressionnants sur la courbe d'entraînement. On remarque également que le modèle semble assez robuste à ces changements, étant capable de retrouver et même baisser la courbe de validation assez rapidement.
Pour observer l'effet sur les paramètres, nous avons observé la courbe des biais supérieurs à -1 dans la Figure 8b ainsi que les poids des pixels 00 et les biais des neurones cachés dans la Figure 8c. On peut alors observer tout d'abord que le nombre de biais supérieurs à -1 augmente logiquement beaucoup dû à la réinitialisation. Cependant le nombre ne semble pas vraiment revenir à la normale, avec des nombres qui restent assez élevés après les réinitialisations, malgré une décroissance. Pour ce qui est des poids synaptiques et des biais neuronaux, il est très intéressant d'observer que pour beaucoup des paramètres réinitialisés, ils suivent ensuite des valeurs très différentes par rapport à avant. Cela semble se rapprocher de ce qui était dit dans la conclusion du papier précédant observant la PCA, qui est que les paramètres suivent une variété différente après coupure dans les paramètres.

4.10 Robustesse aux lésions pour la Maximum Likelihood

Pour observer si on observe bien une différence de robustesse du réseau par rapport à un apprentissage simplement de maximum de vraisemblance, on effectue la même expérience en ne conservant que le terme de vraisemblance (Pas de prior et de stochasticité). On observe les courbes d'erreurs dans la Figure 9a. Celles-ci sont peu différentes de courbes normales, malgré les réinitialisations. Cependant ce cas est très simple et n'est pas un très bon cadre pour démontrer l'utilité du synaptic sampling.
Toutefois il est quand même très intéréssant d'observer quelques valeurs de poids synaptiques (en 20,10) et de biais (cachés) sur la Figure 9b. On peut observer alors l'incroyable uniformité de l'apprentissage obtenu quand le seul terme présent est celui de la vraisemblance. Les paramètres évoluent monotonement (avec des variations très similaires entre eux). De plus, les réinitialisations affectent relativement peu leurs valeurs. En effet, l'évolution semble être toujours la même après, la plupart des termes gardant la même pente, contrairement à ce que nous avons vu précédemment. Le réseau semble peut être rester sur la même variété qu'avant.

4.11 Apprentissage sans stochasticité

Il est également intéressant d'observer l'apprentissage du réseau sans la présence du terme de stochasticité. Le problème principal revient alors à une maximisation du postérieur. On peut observer les courbes d'erreurs dans la Figure 10a. Il ne semble pas il y avoir de changements majeurs dans l'apprentissage sans stochasticité. On observe également la distribution des poids synaptiques dans la Figure 10b. On observe alors que la distribution que l'on retrouve est beaucoup plus propre et homogène que avec la stochasticité. On retrouve un postérieur assez bien défini.
De même en observant par exemple les poids synaptiques en 20,10 sur la Figure 10c. On observe un apprentissage beaucoup plus stable et uniforme que précédemment. Cela confirme que la stochasticité a un impact fort sur l'apprentissage, qui serait beaucoup plus uniforme et déterministe sans ce bruit aléatoire. Pour ce problème simple cela ne pose pas de problème, mais

on comprend pourquoi cela peut causer des pertes de généralité et de l'overfitting dans des cas plus complexes.

5 Conclusion

Dans ce projet, nous avons analysé les articles Synaptic Sampling : A Bayesian Approach to Neural Network Plasticity and Rewiring et Network Plasticity as Bayesian Inference et réimplémenté l'une de leurs expériences et avons poussé l'analyse de ce modèle en détail. Le cadre théorique présenté par ce papier est une évolution logique de l'apprentissage de réseau vers un modèle bayésien, et il propose une dynamique de paramètres de réseau logique et justifiant des résultats biologiques sur le développement des synapses (notamment leur plasticité et leur reconnexions). L'incorporation de ce cadre mathématique dans des modèles neuronaux est assez simple et permet d'obtenir des modèles robustes à des imprévus comme des lésions importantes sur les paramètres du réseau.
Notre analyse de l'implémentation du Synaptic Sampling dans une machine de Boltzmann restreinte permet de justifier quelques résultats de l'article notamment sur la robustesse et la capacité d'adaptation du réseau à des changements. Nous avons pu essayer de nombreuses expériences poussant le modèle dans ses limites et permettant l'analyse de nombreux points de l'entraînement. Nous pensons qu'il est possible d'aller très loin dans l'analyse du synaptic sampling pour des modèles bien plus complexes, et aimerons lire la suite de la recherche des auteurs.

Références

[1] D. Kappel, S. Habenschuss, R. Legenstein, and W. Maass, "Synaptic sampling : A bayesian approach to neural network plasticity and rewiring," in *Proceedings of NIPS*, ., 2015.

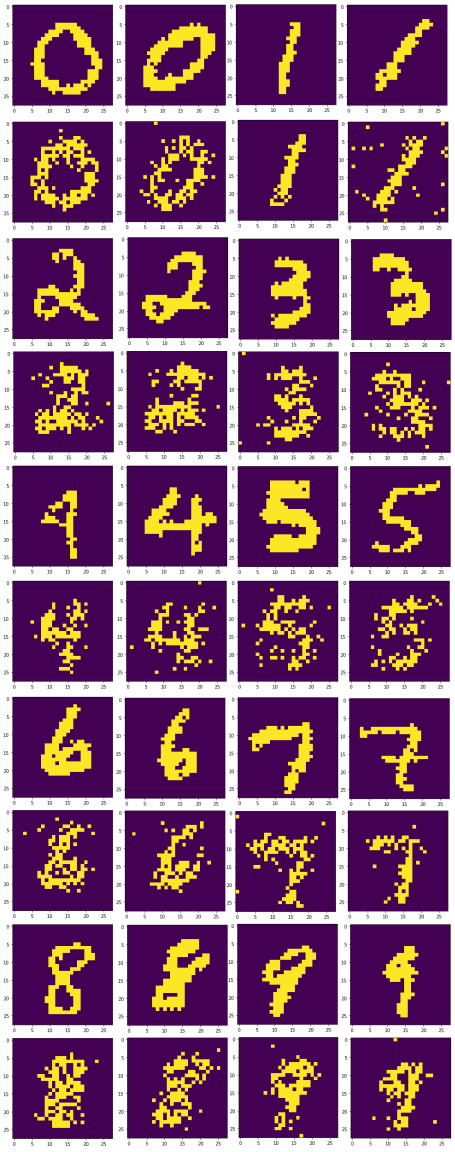
[2] D. Kappel, S. Habenschuss, R. Legenstein, and W. Maass, "Network plasticity as bayesian inference," *PLOS Computational Biology*, vol. 11, 04 2015.

[3] "TensorFlow : Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

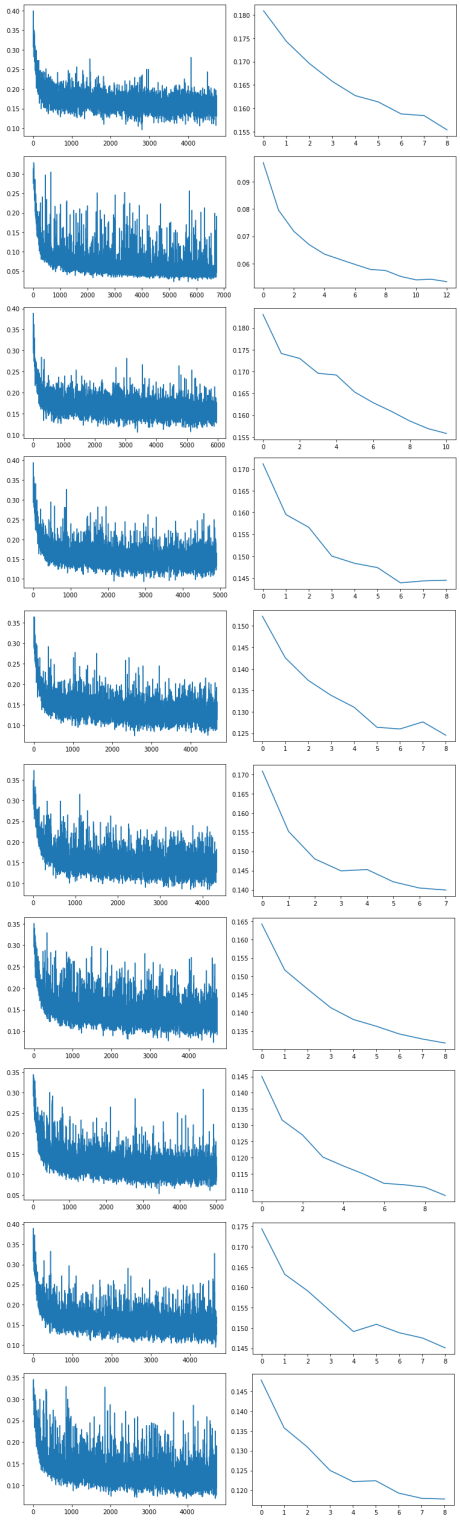
[4] Jakobovski, "Free spoken digit dataset." https://github.com/Jakobovski/free-spoken-digit-dataset, 2019.

[5] mmmaat, "Spectral." https://github.com/bootphon/spectral, 2019.

[6] G. Hinton, "Article training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, pp. 1771–800, 09 2002.



(a) Exemples de résultats pour les 10 labels. Les lignes alternent entre les inputs binarisés et les outputs reconstruits par le réseau.



(b) Courbes d'erreurs pour l'apprentissage des 10 datasets. La colonne de gauche représente l'erreur d'entraînement et celle de droite celle de validation.

FIGURE 1 : Résultats préliminaires sur MNIST.

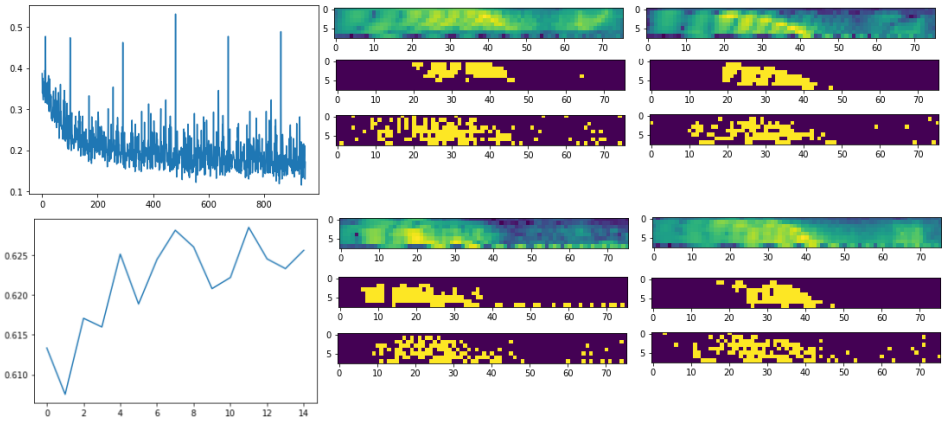


FIGURE 2 : Résultats d'apprentissage de données audio. A gauche la courbe d'erreur d'apprentissage et de validation de haut en bas. A droite 4 exemples d'inputs, d'inputs binarisés, et de reconstruction par le réseau de haut en bas.

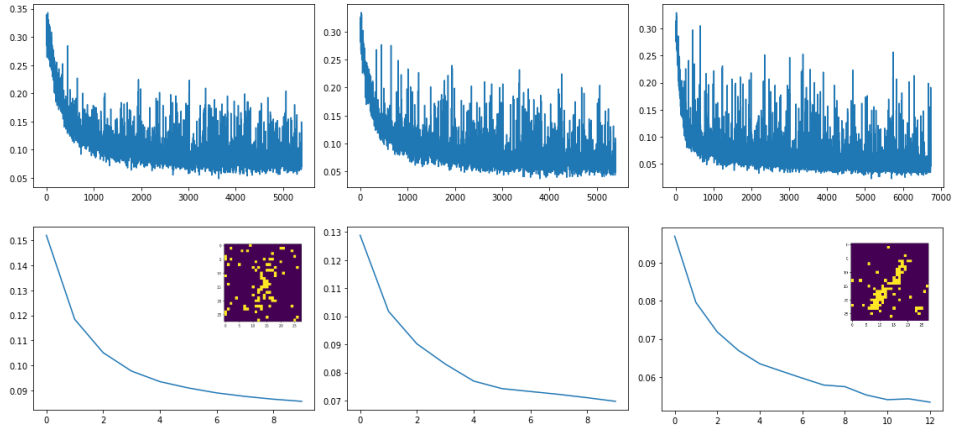
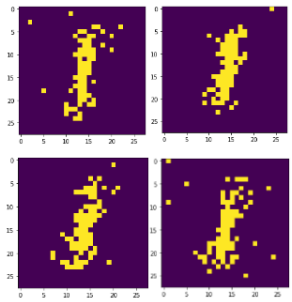
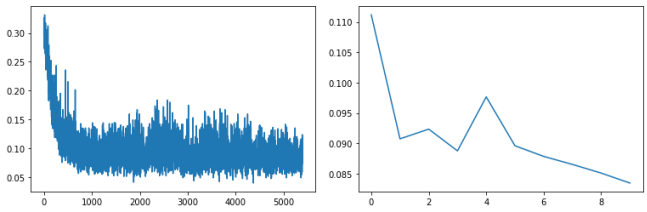


FIGURE 3 : Comparaison de courbes d'erreurs pour un modèle à 1, 3 et 9 neurones cachés. On présente également un exemple de reconstruction du même input pour les deux modèles extrêmes.

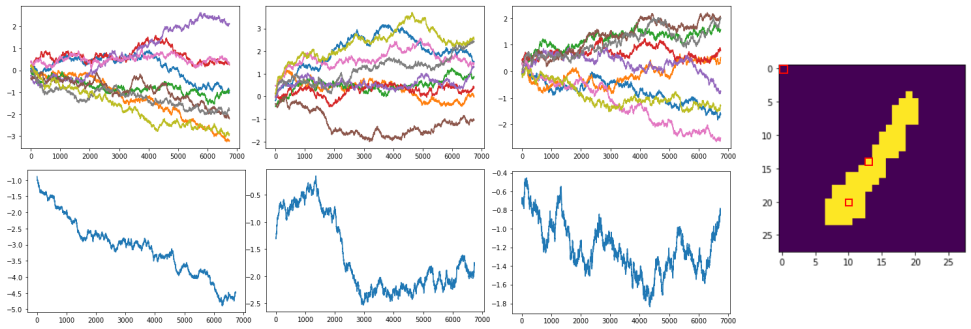


(a) Exemples de 4 reconstructions pour le réseau à double couches cachées.

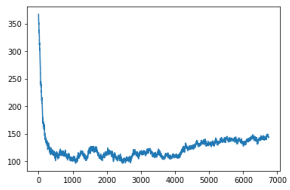


(b) Courbes d'erreurs pour le réseau à double couches cachées.

FIGURE 4 : Résultats de l'ajout d'une deuxième couche cachée au réseau.



(a) Courbe d'évolution des valeurs des poids synaptiques sur la ligne du haut et des biais neuronaux sur la ligne du bas pour les pixels 0,0, 14,14 et 20,10 de gauche à droite. Leur position est représenté sur un input d'exemple.



(b) Evolution du nombre de biais neuronaux supérieurs à -1.

FIGURE 5 : Quelques courbes d'évolution des paramètres.

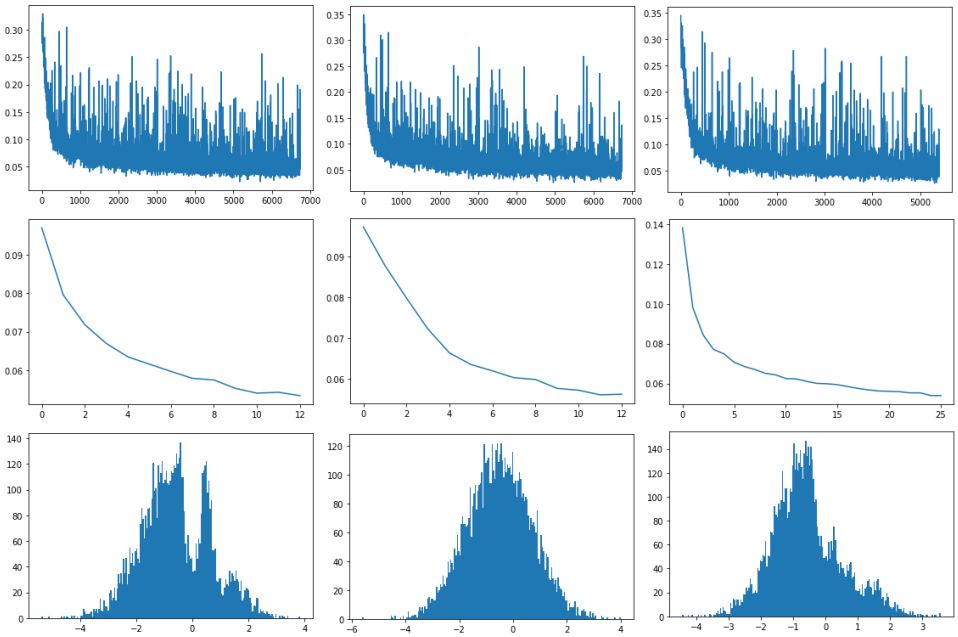
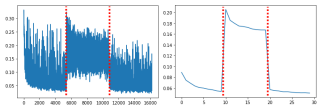
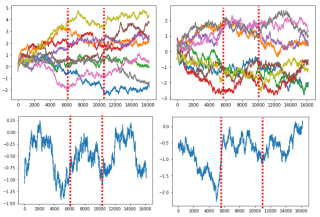


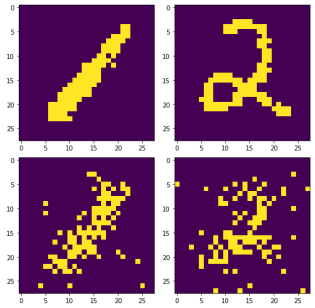
FIGURE 6 : Comparaison de courbes d'erreurs (entraînement et validation) et de la distribution des poids synaptiques pour trois modèles avec des prior différents. De gauche à droite, uniforme, bi-gaussien et tri-gaussien.



(a) Courbe d'erreurs du modèle. Les traits pointillés rouges indiquent un changement de dataset.

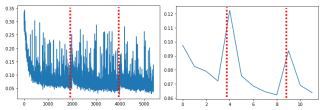


(b) Évolution des poids synaptiques et des biais neuronaux en 14,14 et en 20,10.

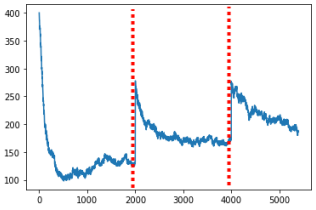


(c) Exemple de reconstruction à la fin du modèle pour un 1 et un 2.

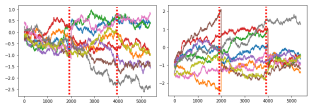
FIGURE 7 : Résultats pour l'entraînement d'un réseau successivement sur le dataset 1, 2, puis 1.



(a) Courbe d'erreurs du modèle. Les traits pointillés rouges indiquent une réinitialisation de la moitié des paramètres.

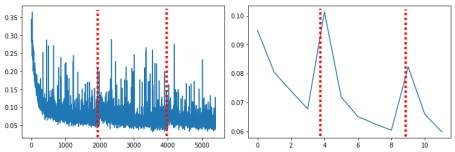


(b) Évolution du nombre de biais supérieurs à -1 pendant l'entraînement.

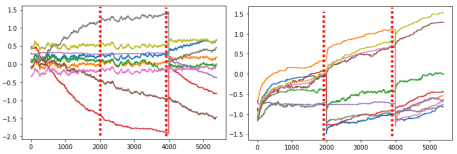


(c) Évolution des poids en 0,0 et des biais des neurones cachés.

FIGURE 8 : Résultats pour l'entraînement d'un réseau dont la moitié des paramètres sont réinitialisés deux fois.

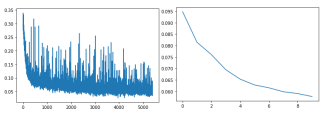


(a) Courbe d'erreurs du modèle.

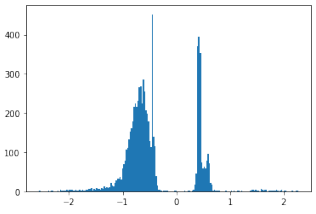


(b) Évolution des poids en 20,10 et des biais des neurones cachés.

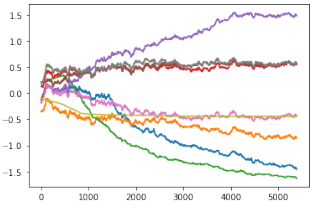
FIGURE 9 : Résultats pour l'entraînement d'un réseau dont la moitié des paramètres sont réinitialisés deux fois, obéissant à la maximisation de la vraisemblance



(a) Courbe d'erreurs du modèle.



(b) Histogramme de la distribution des poids neuronaux.



(c) Évolution des poids en 20,10.

FIGURE 10 : Résultats pour l'entraînement d'un réseau sans le paramètre de stochasticité.