ÉTUDE EMPIRIQUE DE L'APPRENTISSAGE CONTINU AVEC DES TÂCHES STRUCTURELLEMENT LIÉES

Yara El Halawani - Talout Chattah - Mohamed Demes - Taher Lmouden

Encadré par: Massinissa Hamidi

ABSTRACT

L'apprentissage continu permet aux modèles d'apprentissage automatique d'acquérir séquentiellement des connaissances à partir d'un flux de tâches tout en limitant les oublis catastrophiques. Cependant, l'efficacité des algorithmes d'apprentissage continu est influencée par les propriétés des séquences de tâches. Dans cette étude, nous analysons la relation entre l'oubli catastrophique et deux propriétés clés des séquences de tâches : la complexité totale et l'hétérogénéité séquentielle. Grâce aux avancées récentes en modélisation de l'espace de tâches, notamment le cadre Task2Vec, nous quantifions ces propriétés et évaluons leur impact sur les performances d'apprentissage continu par une analyse de corrélation. Nos résultats révèlent une forte corrélation positive entre la complexité totale et l'oubli catastrophique, indiquant que des séquences de tâches plus complexes entraînent des taux d'erreur plus élevés.À l'inverse, nous observons des corrélations faibles, voire négatives, entre l'hétérogénéité séquentielle et les taux d'erreur, suggérant que la dissimilarité des tâches n'aggrave pas nécessairement l'oubli et peut, dans certains cas, améliorer le transfert de connaissances. Ces observations soulignent l'importance d'intégrer la complexité des tâches dans la conception de benchmarks et d'algorithmes d'apprentissage continu. De plus, nos résultats soulignent la nécessité de stratégies adaptatives optimisant le transfert de tâches en fonction de relations spécifiques entre elles. Cette étude fournit un cadre fondamental pour comprendre et améliorer les méthodologies d'apprentissage continu en s'appuyant sur une analyse structurée des séquences de tâches.

Mots clés: Apprentissage continu, complexité totale, hétérogénéité séquentielle, Task2Vec, embeddings.

Yara El Halawani: 20235309@etud.univ-evry.fr
Talout Chattah: 20235542@etud.univ-evry.fr
Mohamed Demes: 20234747@etud.univ-evry.fr
Taher Lmouden: 20235157@etud.univ-evry.fr

TABLE DE MATIÈRES

1 Introduction

L'apprentissage continu est une approche cruciale qui permet aux modèles d'intelligence artificielle d'acquérir séquentiellement de nouvelles connaissances sans oublier les compétences acquises précédemment. Cette capacité est essentielle pour les applications où les données évoluent au fil du temps, telles que la robotique, le traitement du langage naturel et la vision par ordinateur. Cependant, l'un des principaux défis de l'apprentissage continu est l'oubli catastrophique, un phénomène où l'apprentissage d'une nouvelle tâche dégrade considérablement les performances des tâches précédemment acquises.

Pour pallier ce problème, diverses stratégies ont été proposées, notamment les techniques de régularisation des poids, la mémoire épisodique et les approches basées sur la répétition des données ou la génération de pseudo-données. Cependant, les performances des algorithmes d'apprentissage continu dépendent fortement de la structure des séquences de tâches sur lesquelles ils sont entraînés. Certaines séquences peuvent être intrinsèquement plus difficiles à apprendre que d'autres, mais les facteurs influençant cette difficulté restent mal compris.

Dans cette étude, nous menons une analyse approfondie des propriétés des séquences de tâches qui influencent l'oubli catastrophique. Nous nous concentrons sur deux aspects clés : la complexité totale d'une séquence de tâches et son hétérogénéité séquentielle. En exploitant les techniques modernes de modélisation de l'espace de tâches, notamment le framework Task2Vec, nous quantifions ces propriétés et analysons leur corrélation avec les taux d'erreur des algorithmes d'apprentissage continu. Nos résultats indiquent que la complexité totale est fortement corrélée à l'oubli, ce qui signifie que des séquences de tâches plus complexes entraînent une dégradation plus importante des performances. À l'inverse, l'hétérogénéité séquentielle a peu d'impact, voire, dans certains cas, peut réduire l'oubli en facilitant le transfert de connaissances.

2 Jeux de données

Le jeu de données utilisé pour les différentes expériences mises en œuvre est le dataset MNIST, un ensemble de données standard utilisé pour les tâches de classification d'images, principalement destiné à la reconnaissance de chiffres manuscrits. Il contient un total de 60 000 images pour l'entraînement et 10000 images pour le test, chacune étant une image en niveaux de gris de 28x28 pixels. Les étiquettes (labels) associées aux images représentent les chiffres de 0 à 9, couvrant ainsi 10 classes uniques. Ce dataset est largement utilisé pour évaluer les performances des modèles d'apprentissage automatique, en raison de sa simplicité et de sa pertinence pour les tâches de reconnaissance de formes et de classification.

3 Algorithmes d'apprentissage continu

3.1 Synaptic Intelligence (SI)

L'algorithme **Synaptic Intelligence (SI)** est une méthode conçue pour aborder efficacement le problème de l'oubli catastrophique dans l'apprentissage continu. Cette approche repose sur l'idée de protéger

les poids importants pour les tâches précédentes tout en permettant aux poids moins significatifs de s'adapter aux nouvelles tâches. L'algorithme ajuste progressivement l'importance des poids au fur et à mesure de l'apprentissage, offrant ainsi une gestion dynamique et souple de l'oubli.

Lors de l'apprentissage d'une nouvelle tâche, SI commence par calculer l'importance de chaque poids en fonction de son impact sur l'erreur du modèle. Cette importance est estimée à partir des changements de chaque poids au cours de l'apprentissage, et les poids ayant un impact majeur sur la performance du modèle sont considérés comme plus importants et doivent être protégés. Pour mesurer cette importance, on utilise la **trace synaptique** ρ_i, définie par :

$$\rho_i = \sum_{t=1}^{T} \Delta w_i(t) \cdot \Delta w_i(t)$$

où $\Delta w_i(t)$ représente la variation du poids w_i au cours de l'itération t.

Cette mesure permet d'évaluer l'importance relative de chaque poids dans la tâche en cours.

Une des particularités de SI est sa capacité à mettre à jour l'importance des poids de manière **dynamique** tout au long de l'apprentissage, contrairement à des approches statiques. Si un poids devient moins important pour une tâche donnée, son importance diminue, permettant ainsi une meilleure adaptation aux nouvelles informations.

Pour préserver les connaissances acquises, SI introduit une **pénalisation** dans la fonction de perte afin d'empêcher des modifications excessives des poids jugés importants. La fonction de perte globale est donc exprimée par :

$$L = L_{\text{task}} + \lambda \sum_i \rho_i \cdot (w_i - w_i^*)^2$$

où L_{task} est la perte relative à la tâche en cours, ρ_i est l'importance du poids w_i, et w_i^* est sa valeur pour la tâche précédente. Cette régularisation permet ainsi de conserver les connaissances passées tout en adaptant le réseau aux nouvelles tâches.

3.2 Average Gradient Episodic Memory (A-GEM)

L'algorithme **Average Gradient Episodic Memory (A-GEM)** est conçu aussi pour lutter contre l'oubli catastrophique dans l'apprentissage continu. A-GEM utilise une mémoire épisodique pour conserver des exemples des anciennes tâches, tout en apprenant de nouvelles tâches.

Le principe d'A-GEM est de maintenir un ensemble d'exemples des anciennes tâches dans la mémoire, plutôt que de les oublier lors de l'apprentissage des nouvelles tâches. Lors de l'apprentissage d'une nouvelle tâche, l'algorithme vérifie que les gradients calculés pour cette tâche ne perturbent pas les connaissances acquises des anciennes tâches. Pour ce faire, A-GEM impose une contrainte qui garantit que les gradients calculés pour la nouvelle tâche ne vont pas à l'encontre des gradients des anciennes tâches.

L'algorithme fonctionne de la manière suivante :

- **Sélection des exemples** : A-GEM conserve un ensemble d'exemples des anciennes tâches dans la mémoire.
- **Calcul des gradients** : Lors de l'entraînement sur la nouvelle tâche, le modèle calcule les gradients associés à cette tâche.
- **Vérification de la contrainte** : A-GEM impose la contrainte $g \cdot g_p \geq 0$, où g est le gradient de la nouvelle tâche et g_p est le gradient moyen des anciennes tâches. Cette contrainte veille à ce que les gradients de la nouvelle tâche ne perturbent pas les connaissances des tâches précédentes.
- **Mise à jour des poids** : Si la contrainte est respectée, les poids du modèle sont mis à jour pour la nouvelle tâche. Si la contrainte est violée, les gradients sont ajustés pour respecter cette condition.

A-GEM permet ainsi de protéger les connaissances passées tout en s'adaptant aux nouvelles tâches et en économisant de la mémoire par rapport aux méthodes qui stockent tous les exemples de données. Cette approche est donc à la fois efficace et légère pour l'apprentissage continu.

3.3 Elastic Weight Consolidation (EWC)

L'algorithme **Elastic Weight Consolidation (EWC)** est une méthode qui stabilise l'apprentissage d'un réseau neuronal sur plusieurs tâches successives en limitant les modifications des poids critiques pour les tâches précédentes. EWC repose sur une régularisation des poids afin de préserver les connaissances déjà acquises.

L'algorithme fonctionne en trois étapes principales. Tout d'abord, après l'apprentissage d'une tâche, EWC évalue l'importance de chaque paramètre en se basant sur la **matrice d'information de Fisher**. Cette matrice estime à quel point une variation de chaque poids affecte la probabilité de sortie du modèle. Seule sa diagonale est conservée pour approximer l'importance des paramètres. Ensuite, une pénalisation est ajoutée à la fonction de perte lors de l'apprentissage d'une nouvelle tâche afin de minimiser la déviation des poids jugés critiques pour les tâches précédentes. Enfin, l'apprentissage est effectué en prenant en compte cette régularisation, permettant ainsi au modèle de s'adapter à la nouvelle tâche tout en maintenant ses performances sur les précédentes.

La fonction de perte modifiée introduite par EWC est donnée par :

$$\mathcal{L}_{\text{EWC}}(\theta) = \mathcal{L}(\theta) + \frac{\lambda}{2} \sum_i F_i(\theta_i - \theta_i^*)^2$$

où :

- $\mathcal{L}(\theta)$ est la fonction de perte standard de la tâche en cours.
- θ^* représente les valeurs des paramètres après l'apprentissage de la tâche précédente.
- F_i est l'élément diagonal de la matrice d'information de Fisher, qui indique l'importance du poids θ_i.
- λ est un hyperparamètre qui contrôle la force de la régularisation.

Grâce à cette approche, EWC permet de restreindre les modifications des paramètres essentiels aux anciennes tâches tout en laissant une certaine

flexibilité pour s'adapter aux nouvelles données. Ainsi, il garantit un apprentissage progressif sans stockage explicite des exemples des tâches précédentes, tout en préservant la stabilité des connaissances acquises.

3.4 Learning Without Forgetting (LwF)

L'algorithme **Learning Without Forgetting (LwF)** propose une approche différente des méthodes basées sur la régularisation des poids ou sur la contrainte des gradients. Plutôt que de restreindre directement les mises à jour des paramètres, LwF préserve les connaissances acquises en maintenant la cohérence des prédictions entre les anciennes et les nouvelles tâches.

L'entraînement avec LwF suit un processus en deux étapes. Tout d'abord, avant d'apprendre une nouvelle tâche, le modèle initial génère des **prédictions** (ou logits) sur les données de la nouvelle tâche, en conservant ses réponses sur les classes des tâches précédentes. Ensuite, lors de l'entraînement sur la nouvelle tâche, la fonction de perte est modifiée pour inclure un terme qui impose au modèle mis à jour de produire des prédictions similaires à celles du modèle initial pour les anciennes classes.

La fonction de perte utilisée dans LwF est définie comme suit :

$$\mathcal{L}_{\text{LwF}} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{distillation}}$$

où :

- $\mathcal{L}_{\text{task}}$ est la perte classique pour la nouvelle tâche.
- $\mathcal{L}_{\text{distillation}}$ est une perte de distillation qui encourage le modèle à conserver des prédictions cohérentes sur les anciennes tâches.
- λ est un hyperparamètre qui contrôle l'importance de la régularisation.

La perte de distillation est généralement basée sur l'entropie croisée entre les prédictions du modèle avant l'apprentissage de la nouvelle tâche et celles du modèle mis à jour :

$$\mathcal{L}_{\text{distillation}} = -\sum_i q_i \log p_i$$

où q_i représente les probabilités de sortie du modèle initial et p_i celles du modèle après mise à jour. Cette régularisation empêche le modèle d'oublier les anciennes prédictions en le forçant à minimiser l'écart entre ses anciennes réponses et celles après l'apprentissage de la nouvelle tâche.

Grâce à cette approche, LwF permet d'apprendre de nouvelles tâches tout en maintenant la cohérence des prédictions sur les précédentes. Il offre ainsi une alternative aux méthodes qui agissent directement sur les paramètres du réseau, en exploitant une contrainte au niveau des sorties du modèle.

4 Évaluation de la pertinence des tâches

4.1 Task2vec

Task2Vec est un framework utilisé pour générer des embeddings de tâches de classification visuelle sous forme de vecteurs dans un espace vectoriel réel. Cet outil puissant permet d'analyser et de comprendre les relations entre différentes tâches à partir des données, facilitant ainsi l'étude de leur interconnexion et le transfert de connaissances entre elles.

L'exécution de Task2Vec sur un jeu de données produit un embedding, c'est-à-dire une représentation numérique des tâches obtenue via la méthode Task2Vec. Ces embeddings sont des vecteurs de dimension fixe qui codent mathématiquement les propriétés essentielles de l'ensemble de données.

Ces représentations reflètent la structure statistique des données ainsi que leur similarité avec d'autres tâches, ouvrant la voie à diverses applications, telles que :

- Analyse de similarité des tâches: comparaison d'ensembles de données pour identifier les tâches liées.
- Méta-apprentissage: information sur la sélection de modèles ou les stratégies d'apprentissage par transfert.
- Sélection de modèles: sélection d'un extracteur de fonctionnalités ou d'un modèle pré-entraîné le mieux adapté à l'ensemble de données.

Lorsque Task2Vec est exécuté sur un ensemble de données, il utilise un réseau neuronal pré-entraîné pour calculer les intégrations de tâches. Cela implique:

- Extraction de caractéristiques: le réseau traite l'ensemble de données en extrayant des représentations de caractéristiques intermédiaires.
- Matrice d'informations de Fisher (FIM): Task2Vec estime l'importance de chaque paramètre du réseau pour l'ensemble de données à l'aide de la FIM, représentant la sensibilité des sorties du modèle aux changements dans l'ensemble de données.
- Génération des embeddings: les embeddings sont des vecteurs de grande dimension résumant les caractéristiques de l'ensemble de données.

4.2 Complexité totale et hétérogénéité séquentielle

La complexité totale mesure la difficulté globale d'une séquence de tâches en sommant les complexités individuelles de chaque tâche. Elle est définie comme :

$$C(T) = \sum_{i=1}^{k} C(t_i) \tag{1}$$

où :

- $T = (t_1, t_2, ..., t_k)$ est une séquence de k tâches distinctes.
- $C(t)$ représente la complexité d'une tâche t.

La complexité d'une tâche est estimée à l'aide du cadre Task2Vec en mesurant sa distance par rapport à une tâche triviale (un embedding situé à l'origine dans l'espace de Fisher) :

$$C(t) = d(e_t, e_0) \tag{2}$$

où :

- e_t est l'embedding Task2Vec de la tâche t.
- e_0 est l'embedding d'une tâche triviale dand notre cas c'est le MNIST dataset sans rotation.
- $d(\cdot, \cdot)$ est la distance cosinus normalisée, définie comme suit :

$$d(e_1, e_2) = \cos\left(\frac{e_1}{e_1 + e_2}, \frac{e_2}{e_1 + e_2}\right) \tag{3}$$

Ainsi, la complexité totale d'une séquence est la somme de ces distances sur l'ensemble des tâches de la séquence.

Hétérogénéité Séquentielle: L'hétérogénéité séquentielle quantifie la dissimilarité totale entre les tâches consécutives d'une séquence. Elle est définie comme :

$$F(T) = \sum_{i=1}^{k-1} F(t_i, t_{i+1}) \tag{4}$$

où :

- $F(t, t')$ représente la dissimilarité entre deux tâches consécutives t et t'.
- Elle est estimée à l'aide des embeddings Task2Vec comme suit :

$$F(t, t') = d(e_t, e_{t'}) \tag{5}$$

où :

- e_t et $e_{t'}$ sont les embeddings Task2Vec des tâches t et t'.
- La fonction de distance $d(e_t, e_{t'})$ est la distance cosinus normalisée :

$$d(e_1, e_2) = \cos\left(\frac{e_1}{e_1 + e_2}, \frac{e_2}{e_1 + e_2}\right) \tag{6}$$

Ainsi, l'hétérogénéité séquentielle d'une séquence de tâches est la somme de ces distances pour chaque paire de tâches consécutives.

5 Expériences

5.1 Résultats du Task2Vec

Dans le but de quantifier de manière utile les propriétés des tâches (complexité et hétérogénéité), révélant des relations importantes entre ces propriétés et les performances des modèles en apprentissage continu, Task2Vec a été appliqué aux jeux de données de deux manières

différentes. Pour la première, nous avons décidé d'augmenter les angles de 24°, en commençant par MNIST jusqu'à MNIST 336°. Un exemple des résultats de similarité entre ces 15 tâches est représenté par la figure 1. La deuxième méthode consiste à augmenter les angles de 45°, en

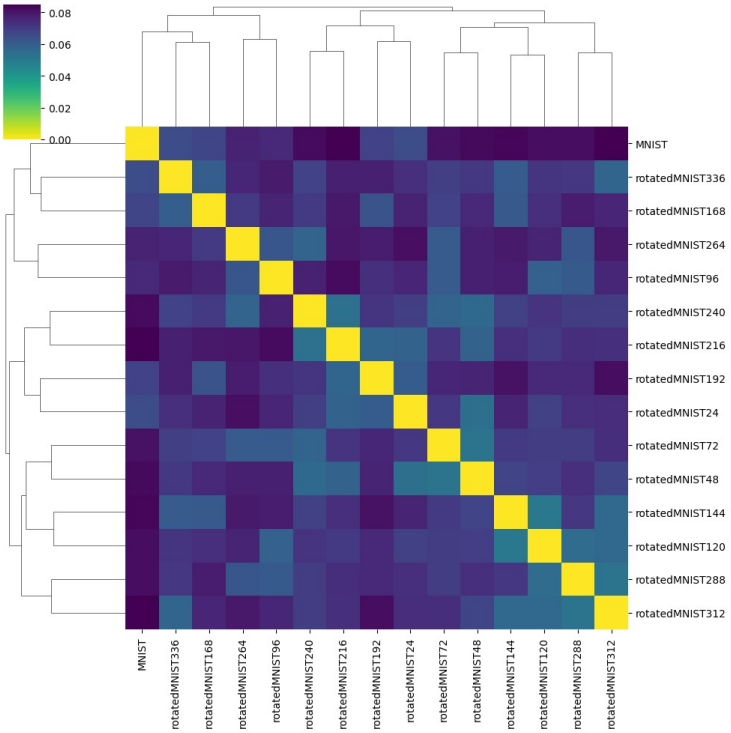

Figure 1: La matrice de similarité de 8 tâches de rotated MNIST.

commençant par MNIST jusqu'à MNIST 270°. Les résultats sont affichés dans la figure 2 ci-dessous. La matrice de similarité mesure la distance ou dissemblance entre les tâches en fonction de leurs embeddings dans l'espace Task2Vec. Les valeurs faibles sont reprensentés par la couleur violet foncé, et indiquent que les tâches sont dissemblables. Tandis que, les valeurs élevées qui sont reprensentés par la couleur jaune, indiquent que les tâches sont similaires, avec peu de variations dans leurs distributions.

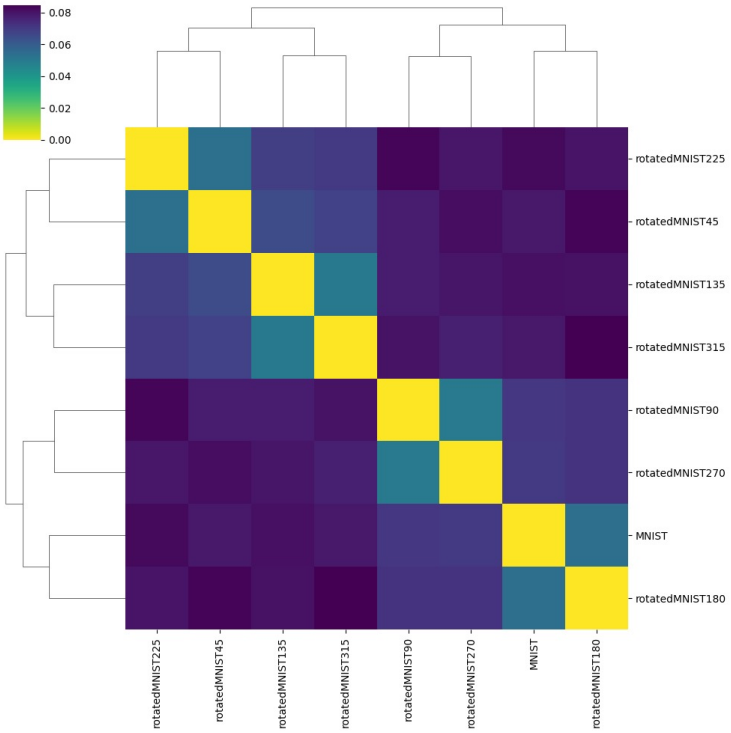

Figure 2: La matrice de similarité de 8 tâches de rotated MNIST.

5.2 Application des algorithmes d'apprentissage continu

Afin d'évaluer la robustesse des algorithmes d'apprentissage continu face à des variations dans l'ordre des tâches, nous avons défini quatre expérimentations distinctes basées sur le dataset Rotated MNIST. Chaque expérimentation repose sur un ensemble de tâches où chaque tâche est une version du dataset avec une rotation spécifique des images.

Dans la première expérimentation, nous avons défini 8 tâches, avec une différence de 45° entre chaque tâche. L'ordre des tâches suit une progression naturelle : la première tâche correspond aux images originales de MNIST (0°), la seconde tâche est obtenue par une rotation de 45°, la troisième par 90°, et ainsi de suite, jusqu'à 315°. Cette configuration permet d'observer la capacité des algorithmes à apprendre progressivement lorsque les transformations sont structurées.

Dans la deuxième expérimentation, nous avons également utilisé 8 tâches avec une différence de 45°, mais cette fois dans un ordre non séquentiel : $1^{\text{ère}}$ tâche (45°), suivie de la $3^{\text{ème}}$ (125°), puis la $0^{\text{ème}}$ (0°), $4^{\text{ème}}$ (180°), $2^{\text{ème}}$ (90°), $7^{\text{ème}}$ (315°), $5^{\text{ème}}$ (225°), et enfin $6^{\text{ème}}$ (270°). Cette approche introduit un changement brusque entre certaines tâches et permet d'analyser comment les modèles réagissent lorsque l'ordre naturel des transformations est perturbé.

Dans la troisième expérimentation, nous avons étendu le nombre de tâches à 15, avec une différence de 24° entre chaque tâche, en conservant un ordre séquentiel : 0°, 24°, 48°, ..., jusqu'à 336°. Cette configuration permet d'analyser l'impact d'un apprentissage plus progressif, où les variations entre les tâches sont plus faibles.

Enfin, dans la quatrième expérimentation, nous avons également utilisé 15 tâches avec une différence de 24°, mais dans un ordre mélangé : $7^{\text{ème}}$ (168°), $3^{\text{ème}}$ (72°), $11^{\text{ème}}$ (264°), $0^{\text{ème}}$ (0°), $14^{\text{ème}}$ (336°), $9^{\text{ème}}$ (216°), $2^{\text{ème}}$ (48°), $10^{\text{ème}}$ (240°), $8^{\text{ème}}$ (192°), $6^{\text{ème}}$ (144°), $5^{\text{ème}}$ (120°), $1^{\text{ère}}$ (24°), $12^{\text{ème}}$ (288°), $13^{\text{ème}}$ (312°), $4^{\text{ème}}$ (96°). Cette configuration permet d'évaluer l'effet de la perturbation de l'ordre des tâches lorsqu'un nombre plus important de transformations est introduit.

Ces quatre expérimentations nous permettent de comparer l'influence du nombre de tâches et de leur ordre sur les performances des algorithmes d'apprentissage continu.

5.2.1 Synaptic Intelligence (SI)

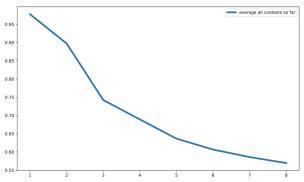

Figure 3: Expérience 1

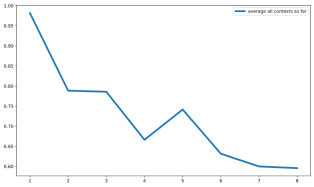

Figure 4: Expérience 2

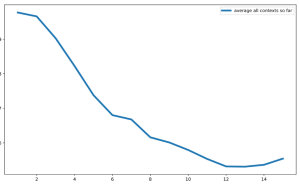

Figure 5: Expérience 3

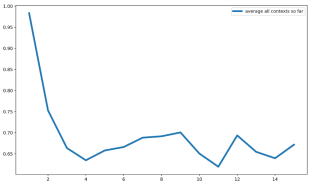

Figure 6: Expérience 4

Figure 7: Accuracy du modèle en utilisant (SI) après chaque task

5.2.2 Average Gradient Episodic Memory (A-GEM)

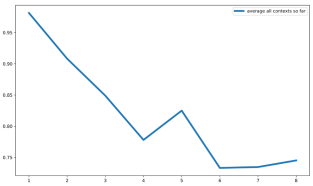

Figure 8: Expérience 1

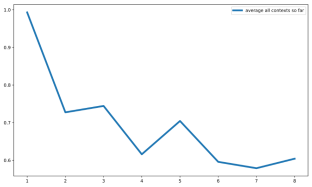

Figure 9: Expérience 2

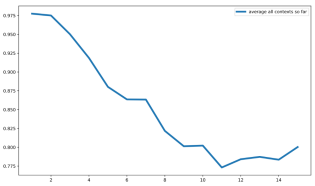

Figure 10: Expérience 3

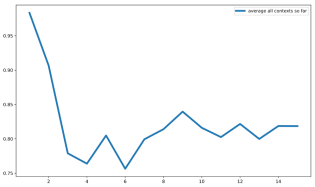

Figure 11: Expérience 4

Figure 12: Accuracy du modèle en utilisant (A-GEM) après chaque task

5.2.3 Elastic Weight Consolidation (EWC)

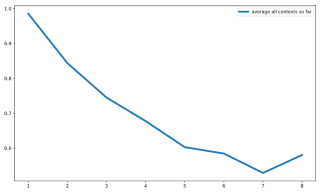

Figure 13: Expérience 1

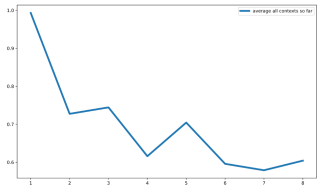

Figure 14: Expérience 2

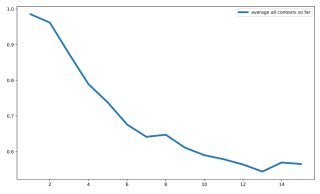

Figure 15: Expérience 3

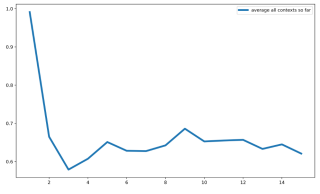

Figure 16: Expérience 4

Figure 17: Accuracy du modèle en utilisant (EWC) après chaque task

5.2.4 Learning Without Forgetting (LwF)

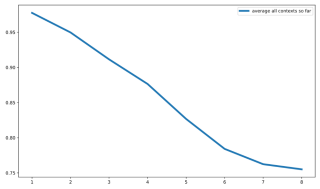

Figure 18: Expérience 1

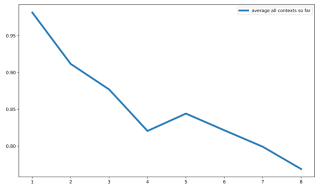

Figure 19: Expérience 2

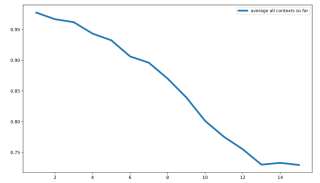

Figure 20: Expérience 3

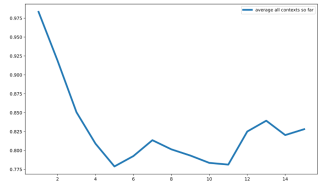

Figure 21: Expérience 4

Figure 22: Accuracy du modèle en utilisant (LwF) après chaque task

D'après les résultats obtenus, nous constatons que parmi les quatre algorithmes testés, **A-GEM (Average Gradient Episodic Memory)** est celui dont la diminution des performances est la plus faible tout au long de l'apprentissage. Cette relative stabilité s'explique par sa capacité à limiter l'impact des nouvelles tâches sur les connaissances précédemment acquises, réduisant ainsi l'oubli progressif observé dans les autres méthodes.

Contrairement aux autres algorithmes, A-GEM montre une meilleure résistance aux perturbations introduites par les nouvelles tâches. Cela lui permet de s'adapter plus efficacement aux variations tout en maintenant un niveau de performance plus élevé par rapport aux autres approches.

Nous observons aussi que le mélange des tâches conduit à de meilleures performances par rapport à un apprentissage séquentiel classique. Dans

les expérimentations 2 et 4, où l'ordre des tâches a été modifié, les algorithmes ont montré une meilleure résistance à l'oubli, suggérant que la diversification des transitions entre les tâches favorise une meilleure généralisation et stabilisation des connaissances acquises.

De plus, nous constatons que l'augmentation du nombre de tâches améliore globalement les performances des modèles. En comparant les expérimentations 1 et 2 (8 tâches) avec les expérimentations 3 et 4 (15 tâches), on remarque que les algorithmes ont mieux préservé leurs performances lorsque le nombre de tâches était plus élevé. Cela peut s'expliquer par le fait qu'un plus grand nombre de tâches introduit des variations plus progressives dans l'apprentissage, permettant ainsi une adaptation plus fluide aux nouvelles données.

Ces observations suggèrent que la structuration des séquences d'apprentissage et le nombre de tâches jouent un rôle clé dans la performance des algorithmes d'apprentissage continu.

5.3 Résultats de Complexité totale et hétérogénéité séquentielle

En utilisant les embeddings obtenus via Task2Vec ainsi que les taux d'accuracies des algorithmes de Continual Learning appliqués à l'ensemble de nos tâches, nous avons généré les graphiques suivants. Ces derniers illustrent la relation entre la complexité totale et le taux d'erreur, ainsi que la corrélation entre l'hétérogénéité séquentielle et le taux d'erreur:

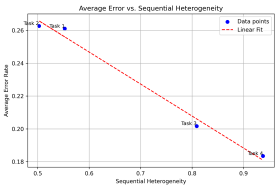

Figure 23: A-GEM

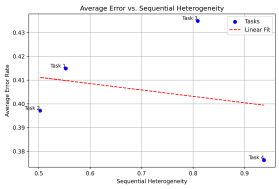

Figure 24: EWC

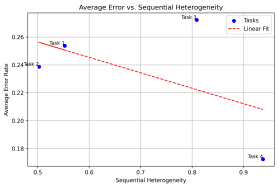

Figure 25: LWF

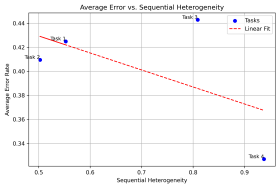

Figure 26: SI

Figure 27: Taux d'erreur VS hétérogénéité séquentielle

- L'hétérogénéité séquentielle réduit les taux d'erreur pour A-GEM et SI mais a peu d'effet sur LwF et EWC.
- Le mélange des tâches (expériences 2 et 4) augmente l'oubli, prouvant que l'ordre des tâches est important.
- Des différences de tâches plus importantes (45° dans les expériences 1 et 2) conduisent à un oubli plus élevé, tandis que des différences plus petites (24° dans les expériences 3 et 4) facilitent la rétention des connaissances.

- L'EWC reste globalement peu affecté par l'hétérogénéité des tâches, confirmant son incapacité à exploiter la diversité des tâches.

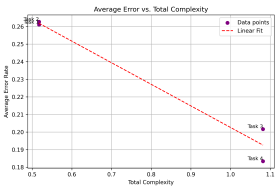

Figure 28: A-GEM

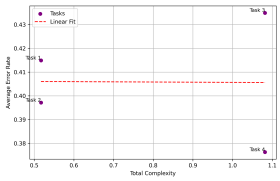

Figure 29: EWC

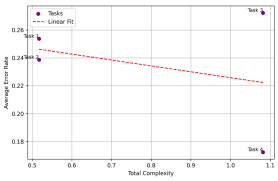

Figure 30: LWF

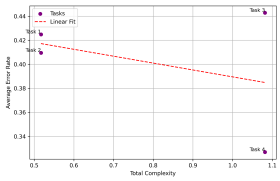

Figure 31: SI

Figure 32: Taux d'erreur VS Complexité Total

- Une complexité plus élevée réduit généralement les taux d'erreur.
- A-GEM présente la meilleure adaptation, tirant parti de la complexité des tâches pour améliorer la rétention.
- EWC ne bénéficie pas beaucoup de la complexité, ce qui suggère qu'il manque de capacité d'apprentissage par transfert.
- LwF rencontre des difficultés, ce qui indique qu'il n'est pas efficace pour s'adapter à des tâches complexes.
- SI s'améliore modérément, gérant la complexité mieux que LwF mais pas aussi bien que A-GEM.

6 Conclusion

Dans ce travail, nous avons étudié l'impact des algorithmes de continual learning sur la dégradation des performances au fil des tâches séquentielles. Nos résultats montrent qu'aucun des algorithmes testés ne permet d'éliminer complètement la dégradation des performances.

Parmi les méthodes expérimentées, A-GEM s'est révélé être le plus performant en maintenant des performances plus stables sur l'ensemble des tâches. Cela s'explique par sa capacité à mieux gérer la rétention des connaissances passées tout en s'adaptant aux nouvelles tâches.

Par ailleurs, l'analyse de la métrique de séquentialité hétérogène nous a permis d'observer un résultat contre-intuitif : contrairement aux attentes, l'ordre des tâches n'a pas nécessairement un impact négatif sur les performances. En effet, dans certains cas, la relation entre certaines tâches peut faciliter la rétention des poids optimaux, favorisant ainsi un transfert de connaissances bénéfique et améliorant les performances globales.

Ces observations ouvrent des perspectives intéressantes pour la conception d'algorithmes de continual learning plus robustes, notamment en explorant des approches qui exploitent intelligemment la relation entre les tâches successives afin de minimiser l'oubli catastrophique.

REFERENCES

7

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6430–6439, 2019.

[2] Abhishek Aich. Elastic weight consolidation (ewc): Nuts and bolts. *arXiv preprint arXiv:2105.04093*, 2021.

[3] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem, 2019. URL `https://arxiv.org/abs/1812.00420`.

[4] Zhizhong Li and Derek Hoiem. Learning without forgetting, 2017. URL `https://arxiv.org/abs/1606.09282`.

[5] Cuong V Nguyen, Alessandro Achille, Michael Lam, Tal Hassner, Vijay Mahadevan, and Stefano Soatto. Toward understanding catastrophic forgetting in continual learning. *arXiv preprint arXiv:1908.01091*, 2019.

[6] Gido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12): 1185–1197, 2022.

[7] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence, 2017. URL `https://arxiv.org/abs/1703.04200`.