# Apprentissage multi-labels et multi-tâches en continu pour données tabulaires: proposition d'un protocole de création de tâches et évaluation de classifieurs

Hugo Peuzet [*,**], Pascale Kuntz [**], Frank Meyer [*], Vincent Lemaire[*], Killian Le Mau[***]

[*]Orange Labs, 2 Avenue Pierre Marzin, 22300 Lannion, prenom.nom@orange.com
[**]LSN de Nantes - Site Polytech - 44300 Nantes, prenom.nom@univ-nantes.fr

**Résumé.** De récents progrès ont été réalisés dans le domaine de la classification multi-labels en flux, où une instance peut être associée à plusieurs labels simultanément. Les recherches les plus récentes se sont concentrées sur l'adaptation des modèles à la distribution dynamique des flux de données non-stationnaires. Cependant, l'apprentissage continu ne se réduit pas à une adaptation à la dérive de concept : des phénomènes tels que l'oubli catastrophique, ainsi que les transferts en avant et en arrière apparaissent lorsque de nouvelles tâches de classification apparaissent dans le flux de données. L'objectif de cet article est d'élaborer un protocole standardisé d'évaluation spécifiquement adapté à l'étude de ces phénomènes pour identifier les stratégies les plus prometteuses pour ce nouveau problème d'apprentissage multi-labels et multi-tâches sur des données tabulaires en flux. Ce protocole inclut (i) la création de flux multi-labels et multi-tâches et (ii) un protocole d'évaluation permettant de mesurer (a) les performances en ligne, (b) les phénomènes liés à l'apprentissage continu et (c) les ressources consommées. Ce protocole est utilisé pour la comparaison de 12 stratégies de classification multi-labels continue sur 4 jeux de données ouvertes de la littérature et 3 jeux de données simulées. Cette analyse exploratoire nous a permis d'identifier le caractère prometteur des réseaux de neurones frugaux couplés à un rejeu de données.

## 1 Introduction

La classification multi-labels (Tsoumakas et Katakis, 2007), qui permet de prédire un sous-ensemble de labels pour une instance décrite par un sous-ensemble d'attributs, a connu un intérêt croissant pour sa capacité à modéliser des situations décisionnelles de façon plus réaliste que la classification binaire. Elle trouve ainsi des applications dans de nombreux cas d'usages (ex : classification de documents, d'images ou plus récemment d'émotions (Tarekegn et al., 2024)). Dans le paradigme classique, les données d'apprentissage sont accessibles dans leur intégralité. Cependant, dans de nouvelles applications, les données sont découvertes au fil du temps et le modèle doit donc être capable de s'adapter. Ces flux peuvent être associés à des volumétries importantes, un débit élevé et des évolutions rapides des caractéristiques des données telles que l'apparition de nouveaux labels ou des changements de distributions statistiques des attributs.

Dans le contexte des flux de données, de nouvelles tâches peuvent apparaître au cours du temps et on parle alors *d'apprentissage continu* (Wang et al., 2024b). Dans ce paradigme, une tâche est un ensemble de données appartenant à un domaine, un espace de sortie ou une distribution spécifique (Hu et al., 2022). Le défi est alors d'apprendre séquentiellement de nouvelles tâches dans des flux non-stationnaires sans oublier totalement les connaissances des tâches précédentes (Van de Ven et al., 2024). Si cet enjeu conduit actuellement à une littérature abondante dans le cadre de l'apprentissage continu, il n'est que depuis peu exploré dans le cadre de la classification multi-labels (Ceccon et al., 2024) et les cas considérés sont essentiellement des images ou des documents textuels. Dans cet article, le domaine applicatif visé est la cybersécurité avec un focus sur la classification de types de menaces sur des données d'usage de l'intranet de grandes entreprises. Les contraintes d'infrastructure et de volumétrie nécessitent des algorithmes d'apprentissage en continu, très réactifs et frugaux. L'article se focalise donc sur les données tabulaires. Les données tabulaires ne pouvant pas être décrites « naturellement » dans des espaces communs de description où l'on peut ré-apprendre des représentations pour améliorer les performances, de nouveaux algorithmes ont été récemment proposés pour la classification multi-labels en flux sur ce type de données (Dalle Pezze et al., 2023) mais ils ne prennent pas en compte de façon explicite les changements de tâches. Par conséquent, à notre connaissance, il n'existe pas encore de protocole d'évaluation standardisé pour évaluer les modèles de classification multi-labels dans des scénarii dynamiques avec des tâches.

L'objectif de cet article est de contribuer à définir un tel protocole afin d'identifier les stratégies les plus prometteuses pour ce nouveau problème. Plus précisément ce protocole doit intégrer deux composantes majeures pour la comparaison expérimentale : (i) la création de flux de données tabulaires non-stationnaires avec tâches associées à des distributions non-stationnaires, et (ii) un ensemble de mesures de performances. La première composante repose en amont sur une modélisation précise du cadre multi-labels en apprentissage continu pour définir notamment les situations de dérive de concept qui, comme l'ont montré récemment (Wang et al., 2024a), différent du cadre classique (Gama et al., 2014). La deuxième composante nécessite l'adaptation de métriques utilisées dans le cadre continu (Díaz-Rodríguez et al., 2018) ou de la classification en flux (Roseberry, 2024). Le développement d'un tel protocole a été motivé par deux ambitions : la mise à disposition d'un cadre expérimental opérationnel pour la communauté travaillant sur la classification multi-labels, et le développement d'un nouvel algorithme « frugal » (Evchenko et al., 2021), c'est-à-dire dans notre cas pouvant s'exécuter sur un ordinateur « standard » sans requêtes vers des ressources externes, afin de garantir une consommation de ressources respectant les contraintes de capacité.

Cet article est organisé en cinq parties principales. La section 2 positionne le problème eu égard à la littérature et propose une formalisation. La section 3 décrit le protocole de création de tâches et d'évaluation. La section 4 présente une expérimentation de ce protocole pour la comparaison de 12 stratégies de classification multi-labels basées sur des principes différents (réseau de neurones et arbres de décisions). Pour finir, la section 5 présente une analyse des résultats obtenus.

## 2 Contexte et concepts

La littérature qui a crû significativement ces dernières années dans le domaine de l'apprentissage continu n'a pas encore une terminologie consensuelle bien stabilisée et il est nécessaire

d'explorer plusieurs paradigmes voisins pour positionner les spécificités de notre problème. On retrouve en effet des objectifs proches en apprentissage incrémental (Yang et al., 2019), en apprentissage tout au long de la vie (Chen et Liu, 2018) et en apprentissage séquentiel (Aljundi et al., 2018).

## 2.1 Etat de l'art

Motivé par les problèmes d'oubli catastrophique des réseaux de neurones (McCloskey et Cohen, 1989), l'apprentissage continu tente de trouver un compromis stabilité-plasticité en exploitant notamment des similarités entre les tâches qui se succèdent dans un flux de données. Deux concepts importants ont été introduits : le « transfert en avant » (resp. « en arrière ») associé à l'influence de l'apprentissage d'une nouvelle expérience d'apprentissage sur des tâches qui n'ont pas encore été apprises (resp. déjà apprises). L'objectif est de maximiser ces deux transferts et d'éviter un transfert en arrière négatif, connu sous le nom d'oubli catastrophique (Lopez-Paz et Ranzato, 2017). Par exemple, en cybersécurité, le modèle doit améliorer ses performances sur les types d'attaque qu'il a appris à détecter lorsqu'il apprend un nouveau type d'attaque (transfert en arrière) et mieux détecter des attaques ultérieures similaires à celles qu'il a déjà appris à détecter, sans avoir besoin d'un apprentissage complet (transfert en avant). Deux types de scenarii se distinguent également : l'apprentissage basé sur les tâches et l'apprentissage sans frontières. Dans le premier, des tâches se succèdent de manière "abrupte" : chaque expérience d'apprentissage (lot de données selon la terminologie de (Van de Ven et al., 2022)) correspond à une tâche différente. Le deuxième autorise des transitions graduelles entre les tâches et il est alors possible que plusieurs tâches se côtoient au sein d'une même expérience d'apprentissage avec une probabilité d'observer chaque tâche changer graduellement au cours du temps (Wang et al., 2022). Le premier scénario reste le plus étudié dans la littérature mais il est soumis à des critiques eu égard aux situations applicatives réelles (Soutif-Cormerais et al., 2023). Le deuxième est considéré comme plus réaliste.

En pratique, la majorité des travaux en apprentissage continu reposent sur un apprentissage hors ligne séquentiel des expériences d'apprentissage avec la possibilité d'effectuer un grand nombre d'itération (Ghunaim et al., 2023). Néanmoins de nouveaux paradigmes ont été proposés pour intégrer des restrictions computationnelles. Ainsi l'apprentissage continu en ligne (Cai et al., 2021) consiste à apprendre en une seule passe sur l'expérience d'apprentissage courante. Une extension plus radicale, appelée apprentissage en ligne sur flux (Gunasekara et al., 2023), consiste à apprendre sur une seul instance de l'expérience d'apprentissage en cours à la fois. Pour les évaluations expérimentales, les jeux de données utilisés sont ceux qui ont été définis dans le cadre de la classification multi-labels classique : issus du dépôt KDIS (Roseberry et al., 2021) ou le jeu de données ALPI (Dalle Pezze et al., 2023). Or, ils ne présentent pas toutes les propriétés requises dans le cadre continu.

## 2.2 Formalisation

La construction d'un scénario spécifique nécessite une formalisation du problème de classification multi-labels continu traité. Nous nous appuyons ici, en les combinant, sur les formalismes introduits en classification multi-labels en flux par (Wang, 2023) et en apprentissage continu par (Van de Ven et al., 2022) et (Cai et al., 2021).

Soit $\mathcal{X} = \mathcal{R}^d$ un espace d'attributs $d$-dimensionnel, $\mathcal{Y} = \{0,1\}^l$ un espace des labels $l$-dimensionnel et un flux de données $F = \{(x_1, y_1), ..., (x_t, y_t), ...\}$ défini par un ensemble, ordonné par un indice temporel $t$, et potentiellement infini de couples $(x_t, y_t)$, où $x_t \in \mathcal{X}$ et $y_t \in \mathcal{Y}$. Dans le cadre multi-labels, une tâche est un ensemble de données associé à une distribution de labels préalablement identifiée. Plus précisément, une tâche $T_n$, associée à une distribution de labels $D_n$ de $\mathcal{Y}$, est un ensemble potentiellement infini de couples $(x_i^n, y_i^n)$ où $x_i^n \in \mathcal{X}$ et $y_i^n \in \mathcal{Y}$, et tels que chaque vecteur de labels $y_i^n = [y_{i,1}^n, ..., y_{i,l}^n]$ se définisse selon la présence/absence des labels : $y_{i,k}^n = 1$ (resp. 0) si le label $\lambda_k$, pour $k = 1$ à $l$, est présent (resp. absent) dans $y_i^n$. Les distributions $D_n$ de labels associées aux tâches sont toutes différentes, et l'on appelle signature de la tâche l'ensemble des labels $S_n$ ayant une probabilité non-nulle d'être présents dans la tâche $T_n$ . Dans la suite, nous notons $\mathcal{T} = \{T_1, T_2, ...\}$ l'ensemble potentiellement infini des tâches. Dans le cadre de notre problème, nous adaptons le concept d'expérience d'apprentissage. Ici, le flux de données $F$ peut être considéré comme un ensemble potentiellement infini d'expériences $\{e_1, e_2, ...\}$ où chaque expérience $e_s = \{(x_{t_i}, y_{t_i}), ..., (x_{t_j}, y_{t_j}), ...\}$ est un sous-ensemble fini de données indexé par un indice temporel $s$.

On peut alors distinguer les deux scenarii d'apprentissage continu. Dans le scénario basé sur les tâches, chaque expérience est un flux fini et stationnaire de données indexées par un indice temporel $s$, $e_s^{T_n} = \{(x_{t_1}, y_{t_1}), ..., (x_{t_j}, y_{t_j}), ...\}$ avec $(x_{t_j}, y_{t_j}) \in T_n$, où chaque exemple est généré selon une distribution de probabilité jointe $\mathbb{P}_s^{T_n}(x, y)$, aussi appelée concept de la tâche $T_n$, inconnue à l'instant $t$. Dans le scenario sans frontières, les expériences sont des flux finis et potentiellement non-stationnaires de données indexées par un indice temporel $s$, $e_s = \{(x_{t_1}^{T_n}, y_{t_1}^{T_n}), ..., (x_{t_j}^{T_m}, y_{t_j}^{T_m}), ...\}$ avec $(x_{t_j}^{T_n}, y_{t_j}^{T_n}) \in T_n$, où chaque exemple est généré selon une distribution de probabilité jointe $\mathbb{P}_t(x, y)$, inconnue à l'instant $t$.

# 3 Protocole de création de flux de données tabulaires avec tâches évolutives

Le protocole présenté dans cet article vise à simuler des flux de données non-stationnaires à partir de jeux de données de données tabulaires à labels multiples. La stratégie consiste à (i) "générer" plusieurs tâches dans le jeu de données, puis (ii) créer un flux de données non-stationnaire en enchaînant des expériences d'apprentissage dans lesquelles les labels et les distributions peuvent évoluer. Dans cet article, nous nous focalisons sur un scénario basé sur les tâches car il permet d'analyser plus explicitement les effets de transferts en apprentissage continu.

• **Génération de tâches.** Le principe repose sur un partitionnement $P$ en $k$ classes d'un jeu classique de données tabulaires initial avec des attributs et labels numériques. Chaque classe $c_i$, $i = 1$ à $k$, est le sous-ensemble de données associé à la tâche $T_i$. Des tests préliminaires avec plusieurs algorithmes de la famille des k-moyennes ont été effectués et leurs résultats ont conduit à la sélection des k-moyennes sphériques (Hornik et al., 2012) appliquées sur les vecteurs de labels. Elles permettent en effet d'obtenir une diversité "pertinente" des distributions des labels dans les classes. Le choix délicat de $k$ est un équilibre à trouver entre la nécessité d'une diversité dans les distributions de label et le risque croissant d'apparition de classes mono-labels quand $k$ augmente. Il a été fixé de manière empirique suite aux tests préliminaires

à $k = 4$. Cette valeur ne garantissant pas la non existence de classes mono-labels, une fusion peut être effectuée dans ce cas avec la classe la plus proche selon la similarité cosinus ; le partitionnement contient alors un nombre $u \leq k$ de tâches.

• **Création et planification du flux.** Étant données $u$ tâches $T_i$, $i = 1$ à $u$, les exemples appartenant à chaque tâche sont partitionnés aléatoirement en trois sous-ensembles : deux expériences d'apprentissages $e_s^{T_i}$ et $e_r^{T_i}$ composées chacune de 35% des données de $T_i$ et un ensemble d'évaluation $v_{T_i}$ composé de 30% des données de $T_i$. L'intérêt des deux expériences d'apprentissage extraites de la même classe est d'étudier le comportement des algorithmes dans le cas de dérive de concept récurrente.

• **Protocole d'évaluation.** La figure 1 présente le processus d'évaluation basée sur deux stratégies différentes permettant d'obtenir des métriques complémentaires. La première est une évaluation en ligne sur les expériences d'apprentissage suivant le protocole "test-then-train" (Losing et al., 2018) où l'algorithme (i) prédit le vecteur de labels d'un nouvel exemple du flux puis (ii) évalue la prédiction, et (iii) apprend avec la vraie valeur du vecteur qui lui est révélée. Le processus est répété sur chaque exemple de l'expérience d'apprentissage dans un ordre de passage séquentiel. La seconde est une évaluation classique en apprentissage continu (Cai et al., 2021), effectuée après le passage complet de chaque expérience d'apprentissage. Le modèle $h_s$ obtenu à la suite de l'apprentissage en flux sur l'expérience d'apprentissage $e_s^{T_i}$ est évalué sur les ensembles d'évaluation $v_{T_j}$ correspondant aux tâches $T_j$, $j = 1$ à $u$. Cette stratégie permet de compléter les mesures classiques (accuracy, F-mesure, etc.) par l'évaluation des transferts avant ou arrière (Díaz-Rodríguez et al., 2018). Notons que l'évaluation n'est réalisée que sur la signature $S_i$ de la tâche à laquelle appartient l'instance en entrée. L'hypothèse sous-jacente est que la tâche à effectuer peut être connue lors de la labélisation de l'instance et que l'on souhaite évaluer le modèle sur cette tâche précise. Des mesures de consommation de ressources (ex : CPU, RAM, durée totale d'exécution, score de frugalité) ont été également intégrées.

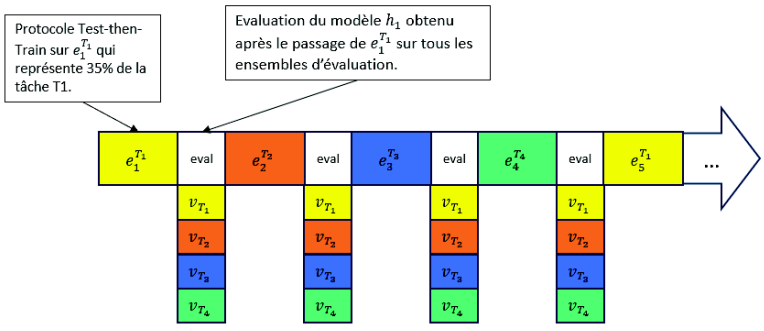


FIG. 1 – *Protocole d'évaluation sur un flux de données dans un scénario basé sur les tâches*

# 4 Évaluations expérimentales

Ce protocole a été implémenté sur la plateforme River (River 0.19.0) (Montiel et al., 2021) qui permet de construire des modèles d'apprentissage en ligne (code disponible sur le dépôt Github Sup (2024)). Les expérimentations ont été réalisées sur une CPU Intel(R) Core(TM) i7-9850H@2.60GHz avec 32 GB de mémoire sur un système Windows 10.

**Stratégies testées et choix des paramètres.**

- Réseaux de neurones [1] : (A1) un réseau de neurones (NN) sans couche cachée équivalent à une régression logistique, (A2) un réseau de neurone sans couche cachée et un calcul de la fonction de perte ciblé sur la signature de la tâche (NN_TL) [2], (A3) un réseau de neurones avec une couche cachée et un calcul de la fonction de perte sur la signature (NN_TLH). S'ajoutent trois variantes (A4-A6) avec du rejeu de données (Buzzega et al., 2021), exécuté en ligne : (i) avec un échantillonage du réservoir (NN_TLH_sampling), (ii) avec une mémoire FIFO (NN_TLH_fifo), (iii) avec un échantillonnage du réservoir et une mémoire FIFO (NN_TLH_memories). Enfin, (A7) un réseau de neurones (NN_TLH_mini_memories) avec une couche cachée, un calcul de fonction de perte ciblé, et un rejeu de données par mini-batch en une seule passe avec échantillonnage du réservoir et une mémoire FIFO.
- Trois méthodes d'adaptation au problème multi-labels basées sur l'arbre de Hoeffding (Hulten et al., 2001) de River (A8-10) : (i) pertinance binaire BR-HT (Zhang et al., 2018), (ii) ensemble taillé LC-HT (Read et al., 2008), (iii) chaîne de classifieurs CC-HT (Read et al., 2009).
- BR-ARF (A11) : algorithme des forêts aléatoires adaptatives (Gomes et al., 2017) de River avec l'adaptation de la pertinence binaire.
- iSOUPtree (Osojnik et al., 2018) (A12) de River.

Les valeurs des hyper-paramètres ont été déterminées par une recherche par quadrillage aléatoire de 10 configurations en utilisant la première expérience d'apprentissage du flux de données (Lee et al., 2024). Leur sélection est basée sur la maximisation du score de frugalité proposé par (Evchenko et al., 2021) pour obtenir un bon compromis consommation / performance.

**Mesures.** Pour le protocole "test-then-train" trois métriques sont mesurées : (i) la macro-averaged balanced accuracy ($BA_{macro}$) associée aux déséquilibres dans la distribution des labels, (ii) l'écart quadratique moyen $RMSE$ associé à la confiance dans les prédictions et (iii) la précision à k ($precision@k$) couramment utilisée dans les milieux industriels.

Après le passage d'une expérience d'apprentissage, le modèle est évalué sur les ensembles d'évaluation de chacune des tâches du flux par trois métriques de la littérature en apprentissage continu : (i) l'accuracy moyenne ($ACC$) (Díaz-Rodríguez et al., 2018) et (ii) le transfert en avant ($FWT$) et (iii) le transfert en arrière ($BWT$). Les transferts sont calculés comme les

1. Une normalisation a été envisagée sur les vecteurs d'attribut en entrée des réseaux de neurones. Cependant, la question de la normalisation en flux étant complexe, et la présence d'une méthode de normalisation ayant eu peu d'impact lors des tests préliminaires, il a été décidé de ne pas en utiliser.

2. Contrairement aux autres algorithmes testés, les réseaux de neurones avec calcul de la fonction de perte ciblé sur la signature de la tâche ont connaissance de la tâche qui est en cours lors de l'apprentissage.

| Jeu de données | Domaine | Exemples | Attributs | Labels | Cardinalité | Diversité |
|---|---|---|---|---|---|---|
| 20NG | Text | 19,300 | 1,006 | 20 | 1.020 | 0.003 |
| Mediamill | Video | 43,907 | 120 | 101 | 4.376 | 0.149 |
| Scene | Image | 2,407 | 294 | 6 | 1.074 | 0.234 |
| Yeast | Biology | 2,417 | 103 | 14 | 4.237 | 0.082 |

TAB. 1 – *Jeux de données du monde réel et leurs caractéristiques*

différences entre la $BA_{macro}$ sur les ensembles d'évaluation des tâches après le passage d'une expérience d'apprentissage et la $BA_{macro}$ avant, avec des valeurs comprises dans $[-1, 1]$.

La frugalité des algorithmes est évaluée par 4 mesures calculées par la librairie python CodeCarbon [3] : le temps d'exécution, l'énergie consommée par la CPU, l'énergie consommée par la RAM, l'énergie totale estimée. Et le compromis performance / frugalité est évalué par la mesure de score de frugalité de (Evchenko et al., 2021), Frug = $\text{ACC}_{final} - \frac{w}{1+\frac{1}{C}}$ avec $C$ la consommation totale estimée durant le passage du flux, et $w = 1$ le poids donné à la frugalité dans notre évaluation.

**Jeux de données.** Quatre flux de données stationnaires du monde réel (texte, image, vidéos, biologie) connus dans la littérature ont été sélectionnés dans le dépôt de jeux de données multi-labels KDIS (voir tableau 1). Et, pour étudier plus finement le comportement des algorithmes, trois jeux de données synthétiques ont été générés par notre protocole : synth_monolab avec des tâches sans labels communs, synth_bilab avec des tâches avec peu de labels communs pour permettre d'observer les phénomènes d'oubli catastrophique, de transfert avant et arrière, et synth_rand avec un scénario plus complexe où les tâches possèdent toutes la même signature de labels.

# 5 Résultats expérimentaux

**Evaluation en ligne :** Pour la majorité des jeux de données et des algorithmes, le passage à l'expérience d'apprentissage d'une tâche qui n'a pas encore été vue introduit sans surprise une chute des performances qui est élevée lorsque la nouvelle tâche qui apparaît dans le flux possède peu ou pas de labels déjà connus. Mais, lorsque des tâches réapparaissent dans le flux, les performances sont plus stables (voir la figure 2). De façon générale, pour l'ensemble des métriques, les réseaux de neurones sont les modèles les plus performants et les stratégies basées sur les arbres, et plus particulièrement LC_HT et les méthodes utilisant la pertinence binaire, sont les moins efficaces.
**Evaluation continue :** Concernant le transfert en arrière $BWT$ moyen, les effets peuvent paraître globalement très faibles en moyenne sur la table 2. Cependant, associé à un $BWT$ à $-0.124$, on observe sans ambiguïté un oubli catastrophique très élevé sur la figure 3. Cela confirme que la quantification du transfert en arrière (resp. celui du transfert en avant) sur la totalité d'un flux reste un problème délicat, notamment lorsque les tâches peuvent réapparaître dans le flux de données. On peut noter des différences entre les algorithmes : les plus promet-

3. Les mesures effectuées par CodeCarbon sur la CPU sont effectuées en utilisant RAPL, une fonctionnalité de Intel qui permet d'estimer assez précisément la consommation d'une CPU. Il est également important de noter que dans un soucis de frugalité, chaque algorithme ne dispose que de 3 heures pour traiter l'intégralité du flux, évaluation et inférence comprise.

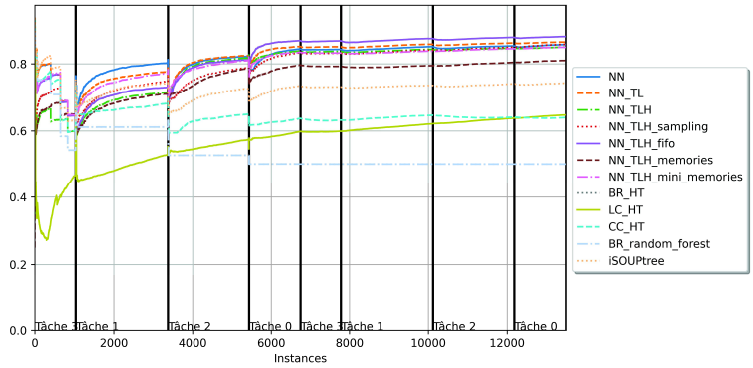


FIG. 2 – *$BA_{macro}$ de 12 stratégies sur le jeu de données 20NG*

teurs semblent être les réseaux de neurones avec un calcul de la fonction de perte ciblé sur la signature de la tâche et du rejeu de mémoire, puis les méthodes basées sur la pertinence binaire. En revanche, les performances de NN, CC_HT et iSOUPtree sont mauvaises (la figure 3 illustre le phénomène d'oubli catastrophique de NN). L'intérêt du calcul de la fonction de perte ciblé sur les labels de la tâche à laquelle appartient l'instance en entrée est confirmé par la figure 4. Il permet en effet de réduire significativement l'oubli catastrophique lorsque les tâches ont peu de labels en commun : la fonction de perte n'étant pas calculée pour les labels qui n'appartiennent pas à la tâche, ces derniers n'interfèrent pas. Notons cependant que ce calcul n'est pas efficace pour toutes les distributions, en particulier pour les données de synth_rand, où toutes les tâches ont la même signature de label, mais où une dérive de concept apparaît à chaque changement de tâche.

Concernant le transfert en avant $FWT$, des différences s'observent entre les différentes stratégies. Les méthodes basées sur la pertinence binaire sont les plus performantes avec un transfert en avant moyen positif, et NN, CC_HT et iSOUPtree semblent également prometteuses. En revanche, les performances des réseaux de neurones avec un calcul de la fonction de perte ciblé et du rejeu de données, et de LC_HT sont mauvaises. La différence de performance entre NN et les réseaux de neurones avec un calcul de la fonction de perte ciblé s'explique par le fait que NN n'apprend pas sur les labels n'appartenant pas à la tâche de l'instance en entrée. Pour l'accuracy moyenne, les réseaux de neurones (resp. les méthodes basées sur les arbres avec de la pertinence binaire) présentent les meilleures (resp. moins bonnes) performances.

**Consommation de ressources :** Le tableau 3 et la figure 5 montrent que les réseaux de neurones offrent le meilleur compromis performance/consommation. On observe également que l'utilisation d'un mini-batch plutôt qu'un rejeu de données en ligne améliore grandement la frugalité malgré une complexité temporelle identique. Ceci s'explique par un effet de bord de l'implémentation (ici l'utilisation de pytorch basé sur du code optimisé en C).

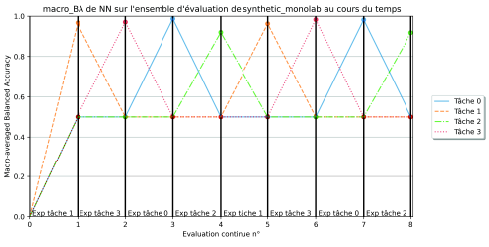


FIG. 3 – *Evolution de la $BA_{macro}$ de NN au cours du temps sur synth_monolab.*

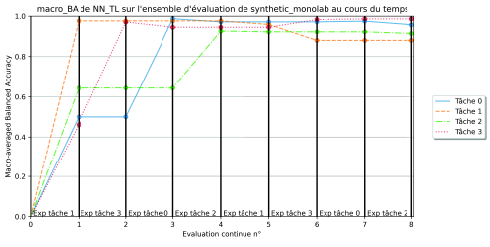


FIG. 4 – *Evolution de la $BA_{macro}$ de NN_TL au cours du temps sur synth_monolab.*

| | 20NG | Mediamill | Scene | Yeast | Synth_monolab | Synth_bilab | Synth_rand | Avg. value | Avg. rank |
|---|---|---|---|---|---|---|---|---|---|
| NN | −0.046 | −0.000 | −0.076 | −0.016 | −0.124 | −0.074 | −0.022 | −0.051 | 10.100 |
| NN_TL | −0.042 | −0.001 | −0.023 | 0.001 | −0.006 | 0.002 | −0.023 | −0.013 | 6.571 |
| NN_TLH | −0.047 | 0.000 | **0.000** | −0.014 | −0.007 | −0.010 | −0.030 | −0.016 | 6.714 |
| NN_TLH_sampling | −0.021 | −0.000 | **0.000** | −0.003 | −0.002 | 0.000 | −0.010 | **−0.005** | 4.143 |
| NN_TLH_fifo | −0.034 | −0.001 | **0.000** | 0.002 | −0.022 | −0.003 | −0.033 | −0.013 | 5.587 |
| NN_TLH_memories | −0.020 | 0.000 | −0.019 | 0.001 | −0.006 | 0.001 | −0.026 | −0.010 | 5.143 |
| NN_TLH_mini_memories | −0.025 | −0.001 | −0.010 | 0.001 | **0.003** | **0.004** | −0.018 | −0.006 | 4.429 |
| BR_HT | **0.010** | −0.003 | −0.006 | 0.000 | −0.066 | −0.036 | −0.007 | −0.015 | 6 |
| LC_HT | −0.003 | 0.000 | −0.022 | **0.006** | −0.091 | −0.070 | −0.018 | −0.028 | 5.429 |
| CC_HT | −0.039 | −0.001 | −0.069 | 0.001 | −0.112 | −0.072 | −0.030 | −0.046 | 9.571 |
| BR_random_forest | 0.000 | **0.001** | **0.000** | 0.002 | −0.077 | −0.039 | **−0.004** | −0.017 | **3.429** |
| iSOUPtree | −0.040 | −0.001 | −0.056 | −0.004 | −0.106 | −0.077 | −0.012 | −0.042 | 9.286 |

TAB. 2 – *Transfert en arrière (BWT) moyen de 12 stratégies sur 7 jeux de données.*

| | 20NG | Mediamill | Scene | Yeast | Synth_monolab | Synth_bilab | Synth_rand | Avg. value | Avg. rank |
|---|---|---|---|---|---|---|---|---|---|
| NN | 0.659 | 0.559 | 0.607 | 0.530 | 0.679 | 0.771 | 0.873 | 0.668 | 6.140 |
| NN_TL | **0.670** | 0.560 | 0.597 | 0.529 | 0.955 | 0.956 | 0.874 | 0.734 | 4.857 |
| NN_TLH | 0.668 | 0.557 | 0.594 | 0.532 | 0.957 | 0.910 | 0.886 | 0.729 | 5.000 |
| NN_TLH_sampling | 0.641 | 0.551 | 0.593 | 0.560 | **0.971** | 0.970 | **0.907** | 0.742 | 4.429 |
| NN_TLH_fifo | 0.658 | 0.522 | 0.594 | **0.573** | 0.931 | **0.974** | 0.897 | 0.735 | 5.429 |
| NN_TLH_memories | 0.658 | 0.533 | 0.691 | 0.571 | 0.954 | 0.968 | 0.904 | 0.754 | 4.143 |
| NN_TLH_mini_memories | 0.667 | **0.561** | **0.706** | 0.570 | 0.965 | 0.972 | 0.904 | **0.764** | **2.143** |
| BR_HT | 0.610 | 0.548 | 0.495 | 0.500 | 0.590 | 0.573 | 0.661 | 0.568 | 10.570 |
| LC_HT | 0.571 | 0.376 | 0.658 | 0.556 | 0.651 | 0.657 | 0.767 | 0.605 | 8.857 |
| CC_HT | 0.611 | 0.540 | 0.615 | 0.538 | 0.667 | 0.813 | 0.903 | 0.669 | 6.714 |
| BR_random_forest | 0.619 | 0.530 | 0.500 | 0.498 | 0.573 | 0.559 | 0.658 | 0.562 | 11.143 |
| iSOUPtree | 0.622 | 0.552 | 0.573 | 0.514 | 0.654 | 0.755 | 0.849 | 0.646 | 8.571 |

TAB. 3 – *Score de frugalité de 12 stratégies sur 7 jeux de données.*

# 6 Conclusion et futurs travaux

Dans cet article, nous avons présenté un nouveau protocole standardisé de création de tâches pour le problème de la classification multi-labels continue en flux de données tabulaires, et nous l'avons mis en œuvre pour évaluer les performances d'un ensemble de stratégies afin de tenter d'identifier les plus prometteuses pour ce problème récent. L'apport du travail présenté est double. Les expérimentations menées constituent une preuve de concept du protocole que nous avons proposé et que nous prévoyons de mettre à disposition de la communauté lorsque nous l'aurons complété notamment par l'introduction de baselines, de tests statistiques pour une comparaison rigoureuse des stratégies et par des mesures complémentaires pour les res-

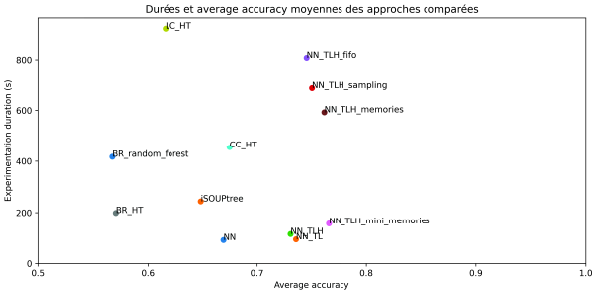


FIG. 5 – *Durée de traitement du flux (corrélée à la consommation estimée) en ordonnée et l'accuracy moyenne en abscisse pour chacune des stratégies comparées.*

sources consommées [4]. Nous prévoyons également de compléter les simulations en intégrant des scenarii sans frontières.

# Références

Aljundi, R., M. Rohrbach, et T. Tuytelaars (2018). Selfless sequential learning. *arXiv preprint arXiv :1806.05421*.

Buzzega, P., M. Boschini, A. Porrello, et S. Calderara (2021). Rethinking experience replay : a bag of tricks for continual learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2180–2187. IEEE.

Cai, Z., O. Sener, et V. Koltun (2021). Online continual learning with natural distribution shifts : An empirical study with visual data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8281–8290.

Ceccon, M., D. D. Pezze, A. Fabris, et G. A. Susto (2024). Multi-label continual learning for the medical domain : A novel benchmark. *arXiv preprint arXiv :2404.06859*.

Chen, Z. et B. Liu (2018). *Lifelong machine learning*, Volume 1. Springer.

Dalle Pezze, D., D. Deronjic, C. Masiero, D. Tosato, A. Beghi, et G. A. Susto (2023). A multi-label continual learning framework to scale deep learning approaches for packaging equipment monitoring. *Engineering Applications of Artificial Intelligence 124*, 106610.

Díaz-Rodríguez, N., V. Lomonaco, D. Filliat, et D. Maltoni (2018). Don't forget, there is more than forgetting : new metrics for continual learning. *arXiv preprint arXiv :1810.13166*.

Evchenko, M., J. Vanschoren, H. H. Hoos, M. Schoenauer, et M. Sebag (2021). Frugal machine learning. *arXiv preprint arXiv :2111.03731*.

Gama, J., I. Žliobaitė, A. Bifet, M. Pechenizkiy, et A. Bouchachia (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR) 46*(4), 1–37.

4. Des résultats et informations complémentaires sont disponibles dans le matériel supplémentaire et le dépôt Github Sup (2024).

Ghunaim, Y., A. Bibi, K. Alhamoud, M. Alfarra, H. A. Al Kader Hammoud, A. Prabhu, P. H. Torr, et B. Ghanem (2023). Real-time evaluation in online continual learning : A new hope. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11888–11897.

Gomes, H. M., A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfharinger, G. Holmes, et T. Abdessalem (2017). Adaptive random forests for evolving data stream classification. *Machine Learning 106*, 1469–1495.

Gunasekara, N., B. Pfahringer, H. M. Gomes, et A. Bifet (2023). Survey on online streaming continual learning. In *International Joint Conference on Artificial Intelligence, IJCAI*.

Hornik, K., I. Feinerer, M. Kober, et C. Buchta (2012). Spherical k-means clustering. *Journal of statistical software 50*, 1–22.

Hu, H., O. Sener, F. Sha, et V. Koltun (2022). Drinking from a firehose : Continual learning with web-scale natural language. *IEEE Transactions on Pattern Analysis and Machine Intelligence 45*(5), 5684–5696.

Hulten, G., L. Spencer, et P. Domingos (2001). Mining time-changing data streams. In *SIGKDD international conference on Knowledge discovery and data mining*, pp. 97–106.

Lee, T. L., S. P. Hellan, L. Ericsson, E. J. Crowley, et A. Storkey (2024). Hyperparameter selection in continual learning. *arXiv preprint arXiv :2404.06466*.

Lopez-Paz, D. et M. Ranzato (2017). Gradient episodic memory for continual learning. *Advances in neural information processing systems 30*.

Losing, V., B. Hammer, et H. Wersing (2018). Incremental on-line learning : A review and comparison of state of the art algorithms. *Neurocomputing 275*, 1261–1274.

McCloskey, M. et N. J. Cohen (1989). Catastrophic interference in connectionist networks : The sequential learning problem. In *Psychology of learning and motivation*, Volume 24, pp. 109–165. Elsevier.

Montiel, J., M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, et al. (2021). River : machine learning for streaming data in python. *Journal of Machine Learning Research 22*(110), 1–8.

Osojnik, A., P. Panov, et S. Džeroski (2018). Tree-based methods for online multi-target regression. *Journal of Intelligent Information Systems 50*, 315–339.

Read, J., B. Pfahringer, et G. Holmes (2008). Multi-label classification using ensembles of pruned sets. In *2008 eighth IEEE international conference on data mining*, pp. 995–1000.

Read, J., B. Pfahringer, G. Holmes, et E. Frank (2009). Classifier chains for multi-label classification. In *European Conference on Machine Learning*, pp. 254–269. Springer.

Roseberry, M. (2024). Adaptive multi-label classification on drifting data streams.

Roseberry, M., B. Krawczyk, Y. Djenouri, et A. Cano (2021). Self-adjusting k nearest neighbors for continual learning from multi-label drifting data streams. *Neurocomputing 442*, 10–25.

Soutif-Cormerais, A., A. Carta, A. Cossu, J. Hurtado, V. Lomonaco, J. Van de Weijer, et H. Hemati (2023). A comprehensive empirical evaluation on online continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3518–3528.

Sup (2024). Matériel supplémentaire et github de l'article "apprentissage multi-label et multi-tâches en continu pour données tabulaires : proposition d'un protocole de création de tâches et évaluation des principaux classifieurs". [Github...] [Résultats supplémentaires...].

Tarekegn, A. N., M. Ullah, et F. A. Cheikh (2024). Deep learning for multi-label learning : A comprehensive survey. *arXiv preprint arXiv :2401.16549*.

Tsoumakas, G. et I. Katakis (2007). Multi-label classification : An overview. *International Journal of Data Warehousing and Mining (IJDWM) 3*(3), 1–13.

Van de Ven, G. M., N. Soures, et D. Kudithipudi (2024). Continual learning and catastrophic forgetting. *arXiv preprint arXiv :2403.05175*.

Van de Ven, G. M., T. Tuytelaars, et A. S. Tolias (2022). Three types of incremental learning. *Nature Machine Intelligence 4*(12), 1185–1197.

Wang, L., X. Zhang, H. Su, et J. Zhu (2024b). A comprehensive survey of continual learning : Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Wang, R., M. Ciccone, G. Luise, A. Yapp, M. Pontil, et C. Ciliberto (2022). Schedule-robust online continual learning. *arXiv preprint arXiv :2210.05561*.

Wang, X. (2023). *Classification Multi-Labels en flux : comparaisons d'approches et nouvelles propositions*. Ph. D. thesis, Nantes Université.

Wang, X., F. Meyer, P. Kuntz, H. Peuzet, et V. Lemaire (2024a). Memory combination-based approaches for multi-label classification on non-stationary data streams. In *Neural Computing and Applications*.

Yang, Q., Y. Gu, et D. Wu (2019). Survey of incremental learning. In *2019 chinese control and decision conference (ccdc)*, pp. 399–404. IEEE.

Zhang, M.-L., Y.-K. Li, X.-Y. Liu, et X. Geng (2018). Binary relevance for multi-label learning : an overview. *Frontiers of Computer Science 12*, 191–202.

# Summary

Recent progress has been made in the field of multi-label stream classification, where an instance can be associated with several labels simultaneously. Most recent research has focused on adapting models to the dynamic distribution of non-stationary data streams. However, continual learning is not reduced to adaptation to concept drift: phenomena such as catastrophic forgetting, forward and backward transfers appear when new classification tasks appear in the data stream. The aim of this article is to develop a standardized evaluation protocol specifically adapted to the study of these phenomena, in order to identify the most promising strategies for this new problem of multi-label, multi-task learning on tabular data in a stream. This protocol includes (i) the creation of multi-label and multi-task streams and (ii) an evaluation protocol to measure (a) online performance, (b) phenomena related to continual learning and (c) resource consumption. This protocol is used to compare 12 continual multi-label classification strategies on 4 open literature datasets and 3 simulated datasets. This exploratory analysis has enabled us to identify the promising nature of frugal neural networks coupled with data replay.