État de l'art : Apprentissage Incrémental

Martin Brami, David Chen, Raphaël Huant, Sébastien Husson, Song Nam Tran

January 30, 2024

Abstract

L'apprentissage incrémental est devenu un nouveau champ de recherche dans le domaine de l'apprentissage automatique. Par rapport au machine learning traditionnel, l'apprentissage incrémental peut continuellement apprendre de nouvelles connaissances à partir de nouveaux échantillons tout en préservant la plupart des connaissances apprises auparavant. Dans cette étude, nous explorons de manière approfondie l'apprentissage incrémental et l'active learning, en les plaçant dans le contexte spécifique de l'analyse des états affectifs humains.

1 Introduction

L'apprentissage incrémental (ou incremental learning) est un concept fondamental dans le domaine de l'intelligence artificielle et de l'apprentissage automatique. Il se réfère à un processus d'apprentissage où un système d'intelligence artificielle est capable d'acquérir de nouvelles connaissances et compétences de manière continue, tout en préservant et en utilisant les connaissances précédemment acquises. L'apprentissage incrémental diffère de l'apprentissage classique en ce sens qu'il ne nécessite pas de reprendre l'ensemble des données d'apprentissage depuis le début à chaque fois qu'un nouveau lot de données est introduit. Au lieu de cela, le système est capable de mettre à jour ses connaissances existantes en intégrant de nouvelles informations de manière graduelle, ce qui permet une adaptation continue aux nouvelles données.

Il existe également de nombreux cas où une classification entièrement objective n'est pas possible, et un avis subjectif, qu'il soit humain ou non, peut se révéler nécessaire. Pour ces données, il est parfois nécessaire d'avoir une intervention humaine pour labelliser celles-ci.

2 Incrémental learning

Pour l'apprentissage incrémental, on peut distinguer 3 types/scénarios possibles [1], "task-incremental learning" ou apprentissage incrémental par tâche, "domain-incremental learning" ou apprentissage incrémental par domaine, "class-incremental learning" ou apprentissage incrémental par classe.

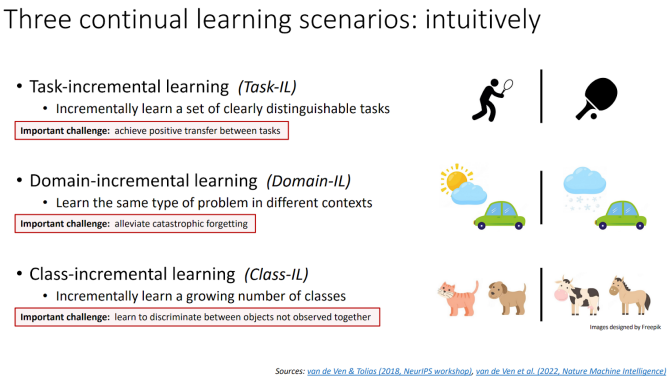

Figure 1: Trois types d'apprentissage incremental.

Dans le premier scénario, apprentissage incrémental par tâche (Task-IL), un algorithme apprend progressivement une série de tâches distinctes. L'algorithme sait quelle tâche doit être accomplie, par une identification explicite de la tâche ou des distinctions entre elles. Cela permet de former des modèles avec des composants spécifiques aux différentes tâches, voire des réseaux complètement distincts pour chaque tâche, évitant ainsi l'oubli. Il permet aussi de partager efficacement les représentations apprises entre les tâches, et de tirer parti des connaissances acquises dans une tâche pour améliorer la performance sur les autres tâches.

L'incrémentation par domaine (Domain-IL) consiste à faire découvrir un nouveau domaine à l'algorithme. La structure du problème reste constante mais le contexte ou la distribution des entrées change, entraînant des décalages de domaine. Similairement au scénario précédent, l'algorithme apprend une série de "tâches" ("domaines" dans ce contexte) et chaque tâche partage les mêmes sorties ou classes possibles. Des exemples de Domain-IL incluent l'apprentissage de la reconnaissance d'objets sous des conditions d'éclairage variables.

Dans le dernier, apprentissage incrémental par classe (Class-IL), l'algorithme doit progressivement apprendre à différencier un nombre croissant d'objets ou de classes. Cet apprentissage se fait par une série de tâches basées sur la classification (appelées "épisodes"), où chaque tâche contient des classes distinctes. L'algorithme doit non seulement être capable de résoudre des tâches individuelles (distinguer les classes au sein d'un épisode), mais aussi identifier à quelle tâche un échantillon appartient (distinguer les classes provenant d'épisodes différents). Il est donc forcé de différencier deux classes sans pour autant avoir eu l'opportunité d'apprendre à les distinguer. [2]

Un concept s'approchant de cette logique qui est déjà bien établi, le transfert learning consiste à utiliser un modèle déjà entraîné à une tâche différente en remplaçant uniquement les dernières couches. Cela peut réduire drastiquement le temps d'entraînement et même permettre une amélioration des performances par rapport à un entraînement de réseau de zéro notamment quand il est question de réseaux à convolutions.

Cette approche incrémentale à un impact sur les architectures de machine learning traditionnelles. Les réseaux de neurones sont fortement affectés par un problème appelé l'oubli catastrophique. Le réseau "remplace" les connaissances acquises dans les précédents batchs par ce qu'il obtient du nouveau. Cela pose problème car les connaissances retenues ne sont pas globales.

Les forêts aléatoires fonctionnent bien. Des méthodes pour les adapter à l'apprentissage continu ont été développées [3] [4] :

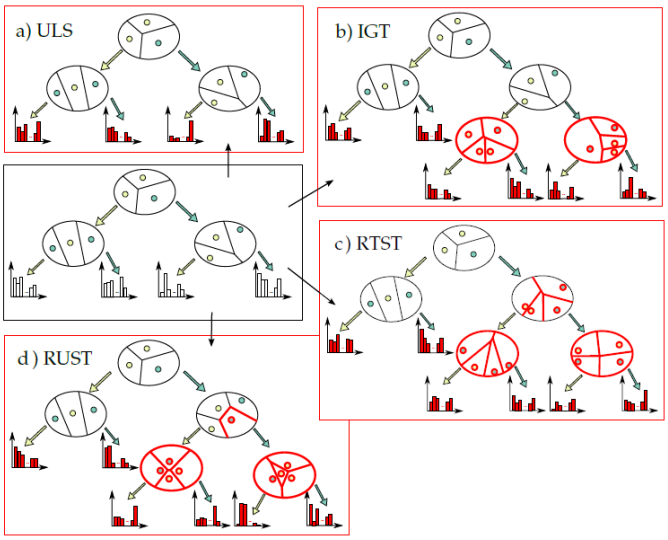

Figure 2: Méthodes pour adapter les forêts aléatoires à l'apprentissage continu.

- Update Leaf Statisitics (ULS) consiste à ne changer que les statistiques de répartition des classes dans les feuilles. La taille de l'arbre et les fonctions de séparations restent inchangées.
- Incrementally Grow Trees (IGT) conserve les feuilles existantes comme telles et ajoute de nouvelles feuilles si assez d'exemples sont disponibles

- Re Train Sub Trees (RTST), des parties de l'arbre sont supprimées et entraînées de nouveau.
- Re Use Sub Trees (RUST) est une déclinaison de RTST, où les feuilles ne sont cette fois-ci pas supprimées, des frontières de décisions sont ajoutées. Cet algorithme est pensé pour être utilisé dans des forêts de NCM. Il suffit ainsi d'ajouter un centroïde supplémentaire.
- Des méthodes qui combinent les forêts à d'autres algorithmes sont aussi proposées comme SVMs et NCM (Nearest Mean Classifier).

3 Concept Drift

Dans la réalité, les propriétés des composants des classes restent rarement les mêmes, ils peuvent changer de manière imprévisible aux cours du temps, et ça a des impacts sur la précision des systèmes et des algorithmes, ce problème est le "Concept Drift" ou la dérive conceptuelle. [5] On peut expliquer ce problème avec "il existe un 't' tel que :

$P_t(X; y) \neq P_{t+1}(X; y)$ avec $P_t(X; y) = P_t(X)P_t(y|X)$ où X est le vecteur de composants et y la classe;

Dépendant des sources de la dérive conceptuel, on peut les catégoriser en plusieurs catégories:

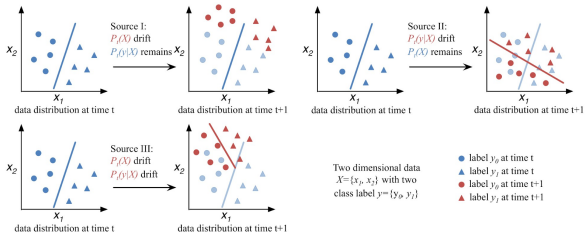

Figure 3: Différentes catégories de dérive conceptuel.

- Dérive virtuelle ("Virtual Drift/ feature space drift"), on a:
  - Source 1: Pt(x) qui change mais Pt(y—X) reste le même
  - La frontière de décision reste la même et la précision n'est pas affecté
- Dérive réelle ("Actual Drift/ decision boundary drift"), on a:
  - Source 2: Pt(y—X) qui change mais Pt(x) reste le même
  - La frontière de décision change et la précision diminue.
- Dérive conceptuelle ("Concept Drift"), on a :
  - Source 3: Pt(x) et Pt(y—X) changent
  - Communique des informations sur l'environnement d'apprentissage

Parmis les dérives conceptuelles, on peut distinguer différents cas:

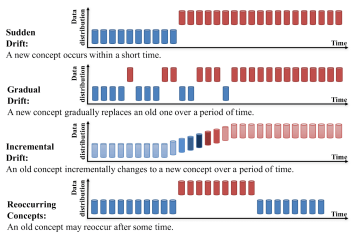

Figure 4: Différentes cas de dérive conceptuel.

- Dérive soudaine (“Sudden Drift”)
- Dérive graduelle (“Gradual Drift”)
- Dérive incrémentale (“Incremental Drift”)
- Concept récurrent(“Reoccuring Concept”)

Pour les 3 premiers, on se concentre sur minimiser la baisse de précision et d’atteindre le taux de récupération le plus rapide lors du processus de transformation de concept. Pour le dernier, on cherche à trouver les concepts historiques les mieux adaptés en un minimum de temps.

Pour pouvoir adapter à ces changements, on a un framework générale:

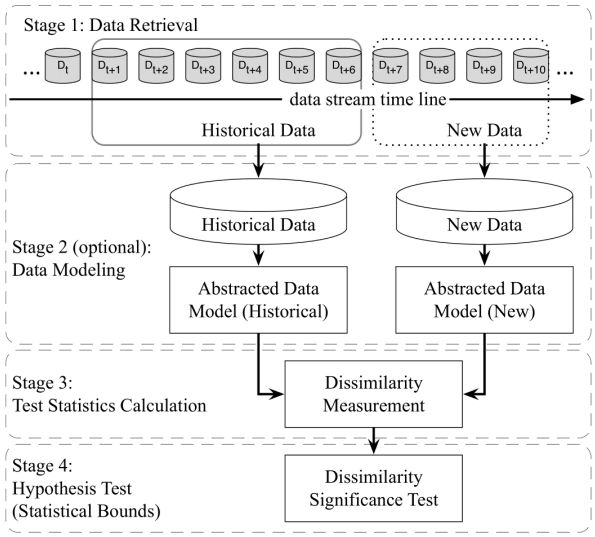

Figure 5: Framework du dérive conceptuel.

- Étape 1 (Récupération de données) : Cette étape consiste à extraire des morceaux de données à partir des flux de données. Étant donné qu’une seule instance de données ne contient pas suffisamment d’informations pour déduire la distribution globale, il est crucial de savoir comment organiser ces morceaux pour en tirer un schéma ou une connaissance significative.
- Étape 2 (Modélisation de données) : L’objectif ici est d’abstraire les données extraites et d’identifier les caractéristiques clés contenant des informations sensibles. Ces caractéristiques sont celles qui auront le plus d’impact sur un système en cas de dérive.
- Étape 3 (Calcul des statistiques de test) : Cette étape concerne la mesure de la dissimilarité ou l’estimation de la distance. Elle permet de quantifier la gravité de la dérive et de générer des statistiques de test pour le test d’hypothèse. C’est l’aspect le plus difficile de la détection de la dérive conceptuelle, car définir une mesure de dissimilarité précise et robuste reste une question ouverte.
- Étape 4 (Test d’Hypothèse) : Cette étape utilise un test d’hypothèse spécifique pour évaluer la signification statistique du changement observé à l’Étape 3. Ces tests permettent de déterminer l’exactitude de la détection de la dérive en établissant les limites statistiques des statistiques de test proposées à l’Étape 3.

Différentes solutions/méthodes sont proposé:

- Les méthodes de détection de dérive basés sur le taux d'erreur ou "Error rate-based drift detection", les algorithmes associés:
  - Lorsqu'une nouvelle donnée est disponible, les algorithmes vérifient si le taux d'erreur global dans une fenêtre temporelle spécifique a significativement augmenté. Si le seuil d'avertissement est atteint, un nouveau modèle est construit tout en utilisant l'ancien pour les prédictions. En cas de dépassement du seuil de dérive, le nouveau modèle remplace l'ancien pour les futures prédictions. Un classificateur est utilisé pour déterminer le taux d'erreur. Les algorithmes fonctionnent de cette manière avec quelques changements.
  - Drift Detection Method (DDM), Learning with Local Drift Detection (LLDD), Early Drift Detection Method (EDDM), Heoffding's inequality based Drift Detection Method (HDDM), Fuzzy Windowing Drift Detection Method (FW-DDM), Dynamic Extreme Learning Machine (DELM)
- Les méthodes de détection de dérive basée sur la distribution des données ou "Data Distribution-based Drift Detection", les algorithmes associés:
  - Il utilisent une fonction ou une métrique de distance pour quantifier la dissimilarité entre la distribution des données historiques et les nouvelles données. Si la dissimilarité est significative, le système déclenche un processus de mise à niveau du modèle. Ces algorithmes abordent le décalage conceptuel à partir de sa source fondamentale, à savoir le décalage de distribution. Ils peuvent identifier avec précision le moment du décalage et aussi fournir des informations sur son emplacement. Cependant, ces algorithmes sont généralement associés à un coût computationnel plus élevé.
  - Relativized Discrepancy (RD), Information-Theoretic Approach (ITA) ou kdqTree, Statistical Change Detection for multidimensional data (SCD),Competence Model-based drift detection (CM).
  - SyncStream, PCA-based change detection framework (PCA-CD), Equal Density Estimation (EDE), Least Squares Density Difference-based Change Detection Test (LSDD-CDT), Incremental version of LSDD-CDT (LSDD-INC), Local Drift Degree-based Density Synchronized Drift Adaptation (LDD-DSDA).
- Les méthodes de détection de dérive par tests d'hypothèses multiples ou "Multiple Hypothesis Test Drift Detection", les algorithmes associés:
  - Les algorithmes utilisent des techniques similaires aux précédentes, ils utilisent des tests d'hypothèses multiples pour détecter le décalage. On peut les diviser en 2 groupes, les tests d'hypothèses multiples parallèles et les tests d'hypothèses multiples hiérarchiques (schéma respective) .

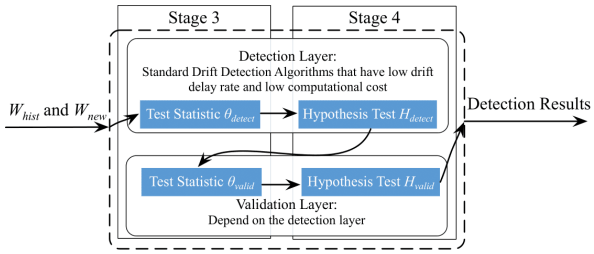

Figure 6: Framework du dérive conceptuel.

  - Les algorithmes associé aux tests d'hypothèses multiples parallèles:

    Just-In-Time adaptive classifiers (JIT), Linear Four Rate drift detection (LFR), Three-layer drift detection (Three-Layer, based on IV-Jac), Ensemble of Detectors (e-Detector), drift detection ensemble (DDE)

– Les algorithmes associé aux tests d'hypothèses multiples hiérarchiques:
Hierarchical Change-Detection Tests (HCDTs), Hierarchical Linear Four Rate (HLFR), Two-Stage Multivariate Shift-Detection based on EWMA (TSMSD-EWMA), Hierarchical Hypothesis Testing with Classification Uncertainty (HHT-CU), Hierarchical Hypothesis Testing with Attribute-wise "Goodness-of-fit" (HHT-AG)

Category	Algorithms	Stage 1	Stage 2	Stage 3	Stage 4
Error rate-based	DDM [20]	Landmark	Learner	Online error rate	Distribution estimation
	EDDM [26]	Landmark	Learner	Online error rate	Distribution estimation
	FW-DDM [5]	Landmark	Learner	Online error rate	Distribution estimation
	DEML [27]	Landmark	Learner	Online error rate	Distribution estimation
	STEPD [30]	Predefined w_{hist}, w_{new}	Learner	Error rate difference	Distribution estimation
	ADWIN [31]	Auto cut w_{hist}, w_{new}	Learner	Error rate difference	Hoeffding's Bound
	ECDD [29]	Landmark	Learner	Online error rate	EWMA Chart
	HDDM [23]	Landmark	Learner	Online error rate	Hoeffding's Bound
	LLDD [25]	Landmark, or sliding w_{hist}, w_{new}	Decision trees	Tree node error rate	Hoeffding's Bound
Data distribution-based	kdqTree [22]	Fixed w_{hist}, Sliding w_{new}	kdqTree	KL divergence	Bootstrapping
	CM [2], [3]	Fixed w_{hist}, Sliding w_{new}	Competence model	Competence distance	Permutation test
	RD [37]	Fixed w_{hist}, Sliding w_{new}	KS structure	Relativized Discrepancy	VC-Dimension
	SCD [38]	Fixed w_{hist}, Sliding w_{new}	kernel density estimator	log-likelihood	Distribution estimation
	EDE [40]	Fixed w_{hist}, Sliding w_{new}	Nearest neighbor	Density scale	Permutation test
	SyncStream [36]	Fixed w_{hist}, Sliding w_{new}	PCA	P-Tree	Wilcoxon test
	PCA-CD [39]	Fixed w_{hist}, Sliding w_{new}	PCA	Change-Score	Page-Hinkley test
	LSDD-CDT [21]	Fixed w_{hist}, Sliding w_{new}	Learner	Relative difference	Distribution estimation
	LSDD-INC [41]	Fixed w_{hist}, Sliding w_{new}	Learner	Relative difference	Distribution estimation
	LDD-DSDA [4]	Fixed w_{hist}, Sliding w_{new}	k-nearest neighbor	Local drift degree	Distribution estimation
Multiple Hypothesis Tests	JIT [19]	Landmark	Selected features	4 configurations	Distribution estimation
	LFR [46]	Landmark	Learner	TP, TN, FP, FN	Distribution estimation
	Three-layer [47]	Sliding both w_{hist}, w_{new}	Learner	$P(y)$, $P(X)$, $P(X\|y)$	Distribution estimation
	e-Detector [48]	depends on base detector	depends	depends	depends
	DDE [49]	depends on base detector	depends	depends	depends
	TSMSD-EWMA [52]	Landmark	Learner	Online error rate	EWMA Chart
	HCDTs [50]	Landmark	Depending on layers	Depending on layers	Depending on layer
	HLFR [51]	Landmark	Learner	TP, TN, FP, FN	Distribution estimation
	HHT-CU [53]	Landmark	Learner	Classification uncertainty	Layer-I Hoeffding's Bound, Layer-II Permutation Test
	HHT-AG [53]	Fixed w_{hist}, Sliding w_{new}	N/A	KS statistic on each attribute	Layer-I KS test, Layer -II 2D KS test

Figure 7: Tableau résumant les différentes méthodologies applicables avec leurs différences entre les étapes du framework.

Dans le tableau suivant, on a les différentes méthodes/algorithmes et à quelle "questions" ils répondent:

Category	Algorithms	When	How	Where
Error rate-based	DDM [20]	√		
	EDDM [26]	√		
	FW-DDM [5]	√		
	DEML [27]	√		
	STEPD [30]	√		
	ADWIN [31]	√		
	ECDD [29]	√		
	HDDM [23]	√		
	LLDD [25]	√		√
Data distribution-based	kdqTree [22]	√	√	√
	CM [2], [3]	√	√	√
	RD [37]	√	√	
	SCD [38]	√	√	
	EDE [40]	√		
	SyncStream [36]	√	√	
	PCA-CD [39]	√	√	
	LSDD-CDT [21]	√		
	LSDD-INC [41]	√		
	LDD-DSDA [4]	√	√	√
Multiple hypothesis tests	JIT [19]	√		
	LFR [46]	√		
	Three-layer drift detection [47]	√		
	e-Detector [48]	√		
	DDE [49]	√		
	EWMA [52]	√		
	HCDTs [50]	√		
	HLFR [51]	√		
	HHT-CU [53]	√		
	HHT-AG [53]	√		

Figure 8: Tableau montrant les différentes algorithmes et à quelle "questions" ils répondent.

- "When" correspond à "Quand" càd le t où la dérive conceptuelle a été détectée.
- "How" correspond à "Comment" càd utiliser une valeur quantifiée pour mesurer la similitude entre le nouveau concept et le concept précédent
- "Where" correspond à "Où" càd déterminer les régions de dérive du changement de concept, les zones de conflit entre un nouveau concept et le concept précédent Ces questions peuvent être répondues différemment dépendant des algorithmes.

Dans tous les cas, le facteur limitant est la quantité de mémoire disponible. Garder la totalité des batchs en mémoire n'est pas une option. L'objectif de l'article [6] est de proposer une architecture qui permette une utilisation efficiente de la mémoire tout en limitant . Ils y décrivent plusieurs solutions contre l'oubli catastrophique. La première consiste à trouver les images les plus représentatives et à les garder en mémoire. C'est une approche assez basique mais qui fonctionne, la difficulté réside cependant à définir quelles images sont les plus discriminantes.

4 Rehearsal et pseudo-rehearsal

Les stratégies de répétition et de pseudo-répétition suivent une idée simple, la rétrospection, pour faire face à l'oubli catastrophique. L'une des raisons de l'oubli catastrophique est que l'apprentissage incrémental ne dispose pas d'une supervision correspondante pour les connaissances antérieures. Si un modèle peut passer en revue les connaissances antérieures lors de l'apprentissage de nouvelles connaissances, il peut atténuer l'oubli catastrophique. [7]

Sur cette base, la méthode de répétition permet au modèle de revoir les anciennes connaissances chaque fois qu'il en apprend de nouvelles en conservant un sous-ensemble des données antérieures.

Alors que la méthode de pseudo-répétition construit un générateur pour apprendre la distribution des données d'entrée. Afin de résoudre le dilemme plasticité-stabilité, lorsque le modèle acquiert de nouvelles connaissances, le générateur produit un lot de pseudo-données dont la distribution est très proche de celle des anciennes données. Lors de la phase de recyclage, le modèle sera supervisé à la fois par les pseudo-données et les nouvelles données.

L'ICaRL a été proposé dans [8], combinant les technologies de distillation des connaissances et de répétition de prototypes. C'est une stratégie d'apprentissage incrémental qui apprend simultanément des classificateurs et une représentation des données.

Pour se faire, ICaRL utilise ces 4 composants principaux :

- Un classificateur basé sur la stratégie des moyenne des exemples les plus proches inspiré de la méthode (Nearest class mean : NCM). Ce nouveau classificateur propre à ICARL est robuste aux changements de la représentation des données et permet de stocker seulement un petit nombre d'exemples par classe en limitant le nombre d'exemples total à une valeur k.
- Une sélection prioritaire des exemples basée sur le herding. qui permet de ne retenir que les exemples les plus pertinents pour chaque classe.
- Une étape d'apprentissage de la représentation avec distillation et répétition. Cette étape permet d'éviter l'oubli catastrophique.
- Une architecture en réseau de neurones convolutionnels (CNN) suivi d'une couche de classification avec autant de nœuds de sortie sigmoïdes que de classes observées, pour l'entraînement et la représentation, ce qui permet de transformer les images d'entrées en vecteur caractéristiques normalisé et de les labelliser.

iCaRL fonctionne comme suit :

- iCaRL utilise un réseau de neurones convolutionnel comme extracteur de caractéristiques, puis classifie les images selon des ensembles d'exemples qu'il sélectionne à partir du flux de données d'entrée. Il rassemble un ensemble d'exemples pour chaque classe observée jusqu'à présent, et s'assure que le nombre total d'images exemplaires ne dépasse jamais la valeur de K. Toutes les classes sont traitées de manière égale, pour t classes observées et K le nombre total d'exemples max, iCaRL utilise m = K/t exemples (arrondis) pour chaque classe.

- Il met à jour la représentation des données, pour cela iCaRL construit un ensemble d'entraînement composé des nouveaux exemples d'entraînements ainsi que des exemples déjà stockés. Ensuite, il évalue le réseau actuel pour chaque exemple et stocke les sorties du réseau pour toutes les classes précédentes. Enfin, il met à jour les paramètres du réseau en minimisant une fonction de perte qui encourage le réseau à donner la classe correcte pour les nouvelles classes (classification loss), et pour les anciennes classes, à reproduire les scores stockés à l'étape précédente (distillation loss).
- Pour prédire un label d'une image, iCaRL calcule un vecteur prototype pour chaque classe observée jusqu'à présent, où le prototype est la moyenne des vecteurs caractéristiques de tous les exemples pour une classe. Il assigne ensuite le label de classe qui a le prototype le plus similaire au vecteur caractéristique de l'image.

Cette méthode présente les avantages et inconvénients suivants.

Les avantages :

- Il est efficace pour apprendre de nouvelles classes tout en conservant les connaissances acquises sur les anciennes classes.
- Il est relativement simple à mettre en œuvre.

Les inconvénients :

- Il nécessite de stocker des exemples, ce qui peut être coûteux en termes de mémoire.
- Il est limité par la taille du budget mémoire alloué aux exemples.
- Il ne peut pas gérer les situations où les classes existantes changent ou disparaissent au fil du temps.

Une option un peu plus avancée proposée par [6] est de conserver les features plutôt que les images en tant que telles. Dans notre cas, garder les AUs et les landmarks pèse moins d'un kilo octet par image, ce qui est plusieurs ordres de magnitude plus petite en termes de stockage. On peut aussi envisager de générer des sets de features à l'aide de GANs.

5 Active learning

En raison des limites de l'approche connexionniste dominante du machine learning, il a souvent du mal avec le raisonnement logique et à identifier les relations de cause à effet. Les chercheurs définissent de nouveaux types d'interactions entre les humains et les algorithmes d'apprentissage automatique, que nous pouvons regrouper sous le terme générique de Human-in-the-loop machine learning (HITL-ML). L'idée n'est pas seulement de rendre l'apprentissage automatique plus précis ou d'obtenir plus rapidement la précision souhaitée, mais aussi d'impliquer l'humain tout au long du processus d'entraînement en lui permettant de superviser la progression du modèle. On se servira de l'approche de l'Active Learning (AL) décrite dans [9].

L'AL est une approche du ML dans laquelle le modèle traite l'humain comme un oracle en étiquetant les données non étiquetées qui ne sont pas claires et qui fournira des informations pertinentes pour le processus d'apprentissage. Le modèle améliore ainsi sa performance en essayant d'utiliser le moins d'exemples labellisés d'apprentissage possible. Il cherche à sélectionner les exemples les plus utiles provenant du dataset non labellisé et le donner à l'oracle. [10] Elle est très efficace dans les contextes où l'on dispose d'un grand nombre de données non étiquetées, mais où la tâche d'annotation est coûteuse et/ou prend beaucoup de temps. Il faut noter que l'oracle n'est pas nécessairement humain, il peut être un autre modèle déjà entraîné; un ensemble de modèle ou la classe la plus choisie est retenu en guise de label.

Dans cette technique, le modèle contrôle les données et interroge une entité ayant une connaissance approfondie du domaine (généralement un expert humain) pour annoter des exemples non étiquetés. Par conséquent, l'AL est une sorte d'apprentissage semi-supervisé puisqu'il utilise à la fois des données étiquetées et non étiquetées.

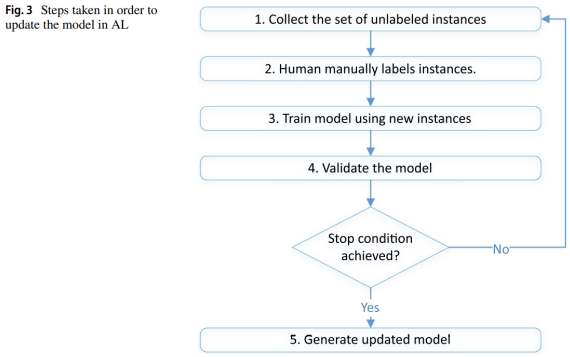

Figure 9: Étapes pour mettre à jour le modèle en active learning

Le processus d'AL commence par la division de l'ensemble des données en deux groupes d'exemples: ceux qui sont étiquetés et ceux qui sont non étiquetés. Ensuite le modèle sélectionne itérativement un nouvel exemple (ou un ensemble d'exemples) dans le groupe non étiqueté et le fournit à l'oracle (expertise humaine) de manière à ce qu'il soit en mesure d'évaluer la qualité des données afin qu'il soit étiqueté. Le système entraîne ensuite le modèle à l'aide des nouvelles données jusqu'à ce que les performances souhaitées ou une condition d'arrêt soit atteinte.

Il faut noter qu'il existe plusieurs méthodes afin de sélectionner la portion et le nombre d'éléments non-labellisés à chaque itération : la méthode aléatoire, incertaines et diverses. La méthode aléatoire est explicite, nous sélectionnons aléatoirement les prochains éléments à classer manuellement. L'échantillonnage par incertitude consiste à sélectionner les éléments les plus durs à classer pour le modèle actuel. Enfin, la sélection par diversité consiste à choisir les éléments les plus éloignés de ceux étant déjà labellisés, afin d'élargir au maximum les données d'entraînement.

Chaque méthode possède des avantages et inconvénients, mais il faut également décider de combien d'éléments ajouter à chaque entraînement. Dans certains cas, il est possible d'actualiser le modèle pour chaque nouvelle classification mais il est souvent préférable de choisir un nombre de nouveaux éléments, que l'on peut baser sur ce que l'on souhaite dépendamment des cas.

L'active learning possède des inconvénients inhérents à la méthode, par exemple le fait qu'il est très coûteux de ré-entraîner un modèle à chaque ajout de labellisation, entraînant la sélection arbitraire d'un nombre d'éléments afin de réduire le temps de calcul. De plus, la qualité de l'entraînement est forcément dépendante de la qualité des labellisations de l'oracle, qui sera toujours supposé exacte.

6 Interactive machine learning

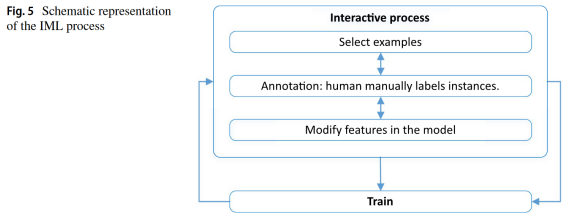

Figure 10: Représentation schématique du processus d'Interactive Machine Learning

L'Interactive Machine Learning (IML) peut être vu comme une généralisation de l'active learning dans le sens ou l'AL ne fait intervenir l'humain que durant le procédé de labellisation tandis que l'IML permet à l'humain de remplir plusieurs rôles si besoin.

La différence principale est que dans le cas de l'AL, le modèle choisi quels éléments il souhaite labellisé et l'oracle répond, tandis qu'ici l'oracle peut également sélectionner quels éléments il souhaite nommer avant l'entraînement. Cette méthode est particulièrement utile dans le cas où très peu de données sont labellisées, rendant la sélection par le modèle peu pertinente.

Bien que conçue pour permettre de créer des modèles se basant sur des données étant difficiles a classées ou ne pouvant être objectivement classées avec certitude, certains juge que cela permet à des non-experts d'entraîner leur modèle sans connaissance approfondie de son fonctionnement.

7 Machine teaching

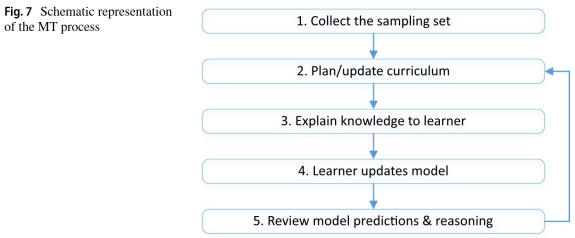

Figure 11: Représentation schématique du processus de Machine Teaching

Comme le nom l'indique le MT implique que l'oracle devient un "learner" et contrôle activement l'avancement de l'apprentissage. Cette intervention peut prendre plusieurs formes, mais il est possible de résumer cela à la sélection de l'apprentissage par le professeur, qu'il soit humain ou machine. En effet, au-delà de choisir quels éléments vont être utilisés pour l'entraînement, le "learner" décide également des labels, et peut également modifier certaines "features" si cela est jugé préférable. C'est également lui qui décide d'arrêter ou non l'entraînement après une itération. Dans le cas où le professeur est non-humain, il faut décider de l'accès que l'on donne à l'algorithme servant de professeur sur l'apprenant. Il est possible de l'autoriser à altérer l'entièreté des paramètres du modèles et des données, ou de faire en sorte qu'il n'accède qu'aux données et aux prédictions de l'élève, par exemple. [11]

L'utilité principale de cette méthode est de permettre à des humains non-experts dans le domaine de créer et paramétrer simplement leur modèle. Tout comme l'active learning et l'interactive machine learning, le machine teaching n'a de réel intérêt que si l'on possède beaucoup de données n'étant pas déjà labellisées. Ces trois méthodes sont similaires dans leurs approches, et l'on peut considérer qu'elles sont des extensions les unes des autres. En effet, l'idée dernière l'AL est de demander l'aide d'un oracle pour l'entraînement, et l'IML renforce ce concept en autorisant l'oracle à intervenir durant l'entraînement, enfin le MT va encore plus loin en permettant l'altération des données et modèles.

Un exemple d'architecture utilisant une approche de type machine teaching est SOLOIST, un modèle de langage devant répondre à des questions utilisateurs et tenir une conversation.

(b) Example snippets for the items compounding the input of SOLOIST model.

Figure 12: Exemple de snippets pour les éléments composant l'entrée du modèle Soloist

Ci-dessus le fonctionnement de SOLOIST, le détail de l'architecture n'étant pas essentiel, nous ne rentrerons pas dans le détail.

Il convient en revanche d'expliquer comment est utilisée le machine teaching dans ce cadre. L'image suivante montre l'interface de "conversation learner", utilisée pour corriger les réponses et les interprétations faîtes par le modèle. Dans ce cas, il ne s'agit donc pas d'influer directement sur le modèle où bien de labelliser des données mais de rectifier les conclusions de ce dernier. Il y a deux étapes durant lesquelles il est possible de corriger le modèle : le "Belief State" et la "Response", le Belief State est l'interprétation que fait le modèle de la conversation en terme de requête, ce qui a été compris de la demande de l'utilisateur, et la Response est la réponse telle que vu par l'utilisateur.

Méthode	Inconvénients	Avantages
DNN	Oubli catastrophique très présent	Nombreux articles sur le sujet
Online SVM	Ne permet pas l'incrémentation par classes	Implémentation dans sklearn
Forêts	Nécessite une réflexion par rapport à la manière d'incrémenter	Forte résistance à l'oubli catastrophique et Peut être couplé à d'autres méthodes
Nearest Class Mean	Approche très simpliste	Le calcul des moyennes permet d'incrémenter de manière efficace et Rapidité
SIGANN's Generator basé sur l'Adversarial Autoencoder	Approche plus complexe et Entraînement exigeant	Génération de données synthétiques réalistes et Prévention de l'Oubli Catastrophique
FearNet	Fragilité par rapport à l' overfitting	Prévention de l'Oubli Catastrophique et Surveille activement les performances du modèle sur les anciennes classes
ICARL	Nécessite de stocker des exemples, ce qui peut être coûteux en termes de mémoire, limité par la taille du budget mémoire alloué aux exemples et ne peut pas gérer les situations où les classes existantes changent ou disparaissent au fil du temps	Efficace pour apprendre de nouvelles classes tout en conservant les connaissances acquises sur les anciennes classes et Relativement simple à mettre en œuvre

Table 1: Tableau récapitulatif des Inconvénients et Avantages des différentes méthodes qu'on a vu.

References

[1] G. M. Van De Ven, T. Tuytelaars, and A. S. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, vol. 4, pp. 1185–1197, Dec. 2022.

[2] D.-W. Zhou, Q.-W. Wang, Z.-H. Qi, H.-J. Ye, D.-C. Zhan, and Z. Liu, "Deep Class-Incremental Learning: A Survey," Feb. 2023.

[3] J. Gonzalez, T. Geoffroy, A. Deshayes, and L. Prevost, "Co-Incrementation: Combining Co-Training and Incremental Learning for Subject-Specific Facial Expression Recognition:," in *Proceedings of the 12th International Conference on Pattern Recognition Applications and Methods*, (Lisbon, Portugal), pp. 270–280, SCITEPRESS - Science and Technology Publications, 2023.

[4] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool, "Incremental Learning of Random Forests for Large-Scale Image Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, pp. 490–503, Mar. 2016.

[5] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under Concept Drift: A Review," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2018.

[6] A. Iscen, J. Zhang, S. Lazebnik, and C. Schmid, "Memory-Efficient Incremental Learning Through Feature Adaptation," Aug. 2020.

[7] Y. Luo, L. Yin, W. Bai, and K. Mao, "An Appraisal of Incremental Learning Methods," *Entropy*, vol. 22, p. 1190, Oct. 2020.

[8] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental Classifier and Representation Learning," Apr. 2017.

[9] E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, and Á. Fernández-Leal, "Human-in-the-loop machine learning: A state of the art," *Artificial Intelligence Review*, vol. 56, pp. 3005–3054, Apr. 2023.

[10] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang, "A Survey of Deep Active Learning," Dec. 2021.

[11] B. Peng, C. Li, J. Li, S. Shayandeh, L. Liden, and J. Gao, "S OLOIST : BuildingTask Bots at Scale with Transfer Learning and Machine Teaching," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 807–824, Aug. 2021.