Mutant-UCB : entre bandits et algorithme évolutionnaire, une approche pour la sélection de modèles

Julie Keisler[1] & Margaux Brégère[2]

[1] *EDF R&D, INRIA Lille Nord Europe - France - julie.keisler@edf.fr*
[2] *EDF R&D, LPSM Sorbonne Université - France - margaux.bregere@edf.fr*

Résumé. Nous présentons la sélection de modèles comme un problème de bandits avec un nombre infini de bras (*infinite-armed bandit*). Les modèles sont les bras du problème de bandit sous-jacent, et le choix d'un bras correspond à un entraînement partiel du modèle (allocation de ressources). La récompense est la performance du modèle sélectionné après son entraînement partiel. Sélectionner le meilleur modèle revient à identifier le meilleur bras (*best-arm identification*). On définit alors le regret comme l'écart entre la performance du modèle optimal et celle du modèle finalement choisi. En partant de cette modélisation, nous proposons un nouvel algorithme, appelé Mutant-UCB, qui généralise l'algorithme UCB-E (développé par Audibert et al. 2010) au problème de bandit stochastique avec un nombre de bras infini, et y incorpore des opérateurs des algorithmes évolutionnaires. Nous avons testé cet algorithme pour optimiser des réseaux de neurones (architectures et hyperparamètres) sur trois jeux de données de classification d'images. Notre algorithme se révèle plus performant que l'état de l'art sur chacun de ces jeux de données ce qui montre que cette approche hybride est pertinente.

Mots-clés. Bandits avec une infinité de bras, sélection de modèles, optimisation d'architectures, optimisation d'hyperparamètres, algorithme évolutionnaire, classification d'images, AutoML, apprentissage en ligne

Abstract. We formulate model selection as an infinite-armed bandit problem. The models are arms, and picking an arm corresponds to a partial training of the model (resource allocation). The reward is the accuracy of the selected model after its partial training. In this best arm identification problem, regret is the gap between the expected accuracy of the optimal model and that of the model finally chosen. We introduce the algorithm Mutant-UCB, a generalization of UCB-E (see Audibert et al. 2010) to the stochastic infinite-armed bandit problem that incorporates operators from evolutionary algorithms. Tests were carried out on the optimization of deep neural networks (architectures and hyperparameters). The results, which outperform the state-of-the-art on three image classification datasets, demonstrate the relevance of our approach.

Keywords. Infinite-armed bandits, Model selection, Neural architecture optimisation, Hyperparameter optimisation, Evolutionnary algorithm, Image classification, AutoML, Online Learning

1 Introduction

Les performances des modèles de *machine learning* dépendent d'un grand nombre de paramètres qui ne peuvent pas être optimisés durant l'entraînement. Tester à la main toutes les combinaisons de ces paramètres est impossible. C'est pourquoi, les techniques permettant de sélectionner automatiquement ces paramètres, regroupées sous le nom d'AutoML (pour *Automated Machine Learning*), ont gagné en popularité (voir Hutter et al. 2019 pour un livre sur le sujet). Nous abordons l'optimisation de ces paramètres, aussi appelée sélection de modèles, d'une manière très générique. Nous ne nous restreignons à aucun type de paramètres à optimiser, qui peuvent être le modèle en lui-même, l'architecture d'un réseau de neurones ou les hyper-paramètres d'une forêt aléatoire par exemple. Notre but est de trouver le meilleur modèle pour une tâche donnée, sans faire d'hypothèse sur la tâche, le type de modèle ou la fonction de récompense à maximiser. Nous supposons uniquement que nous avons accès à un nombre infini de modèles potentiels. Pour réaliser notre optimisation, nous nous fixons un budget de T ressources à utiliser pour entraîner nos modèles. Ces ressources sont allouées sous la forme d'entraînements partiels. Le modèle final est choisi en s'appuyant sur un compromis entre une bonne exploration (entraîner un grand nombre de modèles) et une bonne exploitation (allouer un budget de ressources conséquent aux modèles prometteurs). Nous nous plaçons dans le cadre des bandits avec une infinité de bras pour gérer ce compromis exploration-exploitation (voir Lattimore and Szepesvári 2020 pour une revue complète).

Dans ce papier, nous considérons ainsi la sélection de modèles comme l'identification du meilleur bras dans le cadre bandit avec un nombre infini de bras. Nous proposons de résoudre ce problème en considérant les algorithmes de type UCB (*Upper Confidence Bounds*, voir Auer et al. 2002). À partir de ces approches, nous avons créé un nouvel algorithme de sélection de modèles, appelé Mutant-UCB, qui utilise un opérateur de mutation venant des algorithmes évolutionnaires. Cet opérateur génère un nouveau modèle appartenant au voisinage du bras sélectionné par l'algorithme de bandits. Mutant-UCB ne fait aucune hypothèse sur la manière d'encoder les modèles à sélectionner (aussi appelé espace de recherche) ou sur la fonction de récompense à maximiser. Il est ainsi applicable à un très grand nombre de configurations. La combinaison d'un algorithme de type UCB avec une allocation adaptative du budget T permet une bonne exploration de l'espace de recherche, tandis que l'opérateur de mutation oriente de manière efficace la recherche vers les solutions prometteuses. Nos résultats sur l'optimisation de réseaux de neurones démontrent l'intérêt de notre approche. La Section 2 présente le cadre bandit pour la sélection de modèles et nous positionne vis-à-vis de l'état de l'art. Dans la Section 3, nous détaillons notre algorithme Mutant-UCB. La Section 4 est dédiée à nos expérimentations sur l'optimisation de réseaux de neurones. Nous validons les performances de Mutant-UCB sur trois jeux de données de classification d'images disponibles en *open-source*. Enfin, la Section 5 discute des avantages de Mutant-UCB par rapport à l'état de l'art et ouvre de nouvelles perspectives de recherche.

2 Modélisation d'un problème de sélection de modèles : une approche bandit

2.1 État de l'art

Les premières stratégies pour la sélection de modèles sont la recherche en grille ou la recherche aléatoire (*Grid Search* et *Random Search*). Des méthodes plus sophistiquées abordent la sélection de modèles comme un problème d'apprentissage séquentiel. Deux approches se distinguent : la **sélection de configuration** (*configuration selection*) pour laquelle de nouveaux modèles "proches" des modèles prometteurs sont choisis séquentiellement, et l'**évaluation de configuration** (*configuration evaluation*, voir Li et al. 2018) qui alloue plus de ressources (temps d'entraînement) aux modèles prometteurs. La première approche suggère l'existence d'une distance entre les modèles (possiblement dans un espace latent sous-jacent), de sorte que deux modèles proches l'un de l'autre auront des performances similaires. La seconde approche, elle, n'introduit aucune notion de proximité entre les modèles, et, de fait, ne repose sur aucune hypothèse concernant leur performance.

Algorithmes évolutionnaires. Parmi les méthodes de sélection de modèles, les algorithmes évolutionnaires sont populaires depuis de nombreuses années (voir par exemple Young et al., 2015). Ils partent d'un ensemble de modèles initiaux, et les font évoluer vers des modèles performants en utilisant des opérateurs unitaires comme la mutation (peu de changement dans la configuration), ou impliquant plusieurs modèles comme le *crossover* (Strumberger et al., 2019). L'un des inconvénients des algorithmes évolutionnaires est le grand nombre de paramètres impliqués, tels que la taille de la population, la fonction de sélection ou le taux d'élitisme. Le choix des valeurs appropriées pour ces paramètres peut s'avérer complexe.

Optimisation bayésienne. L'optimisation bayésienne s'est récemment imposée comme une approche plus efficace que les méthodes évolutionnaires pour l'AutoML (voir, entre autres, Zoph and Le 2016). Cette technique d'optimisation séquentielle, utilisée pour la minimisation de fonctions boîtes noires repose sur deux composantes principales, un modèle de substitution (*surrogate model*) qui permet d'approximer la fonction objectif inconnue et une fonction d'acquisition qui permet de choisir le prochain élément de l'espace de recherche à évaluer. Les fonctions d'acquisition nécessitent un espace de recherche numérique voir continu (Garrido-Merchán and Hernández-Lobato, 2020). C'est pourquoi nous n'utilisons pas d'optimisation bayésienne dans nos expériences, car nous souhaitons un cadre plus générique.

Algorithmes de bandits. Toujours dans le domaine de l'optimisation bayésienne, des extensions de l'algorithme UCB classique ont été utilisés pour l'optimisation et la sélection de modèles tel que GP-UCB et KernelUCB (voir respectivement Srinivas et al. 2009 et Valko et al. 2013). Les méthodes d'évaluation de configurations ont également été étudiées dans un cadre de bandit à bras multiples ou infinis (*infinite* ou *multi-armed bandit*). À chaque itération de l'algorithme, un nouveau bras/modèle peut être tiré dans un espace de recherche infini contenant les modèles puis être ajouté à l'ensemble des modèles déjà (plus ou moins) entraînés. L'algorithme de *Sequential* (ou *Successive*) *Halving*, répartit le budget donné entre un certain

nombre (supposé optimal) de tours d'élimination, et tire de manière uniforme au sein d'un tour les bras à évaluer. Il a été étendu par Li et al. 2018, qui propose Hyperband, une extension plus robuste utilisée pour l'optimisation d'hyperparamètres de réseaux de neurones. Enfin, des méthodes hybrides combinant la sélection et l'évaluation de configurations adaptatives ont également été proposées par Kandasamy et al. 2016 qui étend GP-UCB pour permettre l'apprentissage séquentiel des modèles et l'allocation de ressources.

2.2 Contributions

Nous proposons un nouvel algorithme de sélection de modèles, Mutant-UCB, qui généralise l'algorithme UCB-E (Audibert et al., 2010) au problème de bandits avec une infinité de bras et incorpore un opérateur de mutation issu des algorithmes évolutionnaires. Notre algorithme combine les approches d'évaluation et de sélection des configurations : il repose sur une stratégie séquentielle et choisit un modèle grâce à un critère basé sur l'UCB. Ensuite, soit il continue l'entraînement du modèle sélectionné (allocation de ressources), soit il crée un nouveau modèle avec la mutation et l'entraîne. L'intuition est que la performance du "modèle mutant" sera proche de celle du modèle original. Bien que des approches bandit aient été utilisées pour l'opérateur de "sélection" d'un algorithme évolutionnaire (voir Li et al., 2013), c'est à notre connaissance la première fois qu'une mutation est incorporée dans un algorithme de bandit. Nous comparons les performances de Mutant-UCB avec une recherche aléatoire, un algorithme évolutionnaire et l'algorithme Hyperband (voir Li et al., 2018) sur trois jeux de données open source collectés pour la classification d'images. Pour une comparaison équitable, Mutant-UCB et l'algorithme évolutionnaire partagent le même opérateur de mutation.

2.3 Modélisation

À chaque itération $t \leq T$ de notre algorithme, un nouveau bras k est choisi dans l'espace de recherche ou créé par mutation. Il est entraîné sur un jeu d'entraînement $\mathcal{D}_{\text{train}}$ puis évalué sur un jeu de validation $\mathcal{D}_{\text{valid}}$ à l'aide d'une fonction de précision (*accuracy*) acc qui correspond à la récompense a_t que nous cherchons à maximiser. Dans ce qui suit, nous appelons f_k un modèle non-entraîné associé au bras k, et $f_k^{N_k}$ lorsqu'il a été entraîné N_k fois.

3 Mutant-UCB

L'Algorithme 1 détaille le pseudo-code de Mutant-UCB. Il échantillonne dans l'espace de recherche K modèles initiaux $f_1, \ldots, f_K$, avec K à fixer. Les K modèles sont sous-entraînés une première fois donnant les précisions $a_k = \text{acc}(f_k^1, \mathcal{D}_{valid})$. Lors des itérations suivantes $t \leq T$, un même bras k pourra être entraîné plusieurs fois, nous permettant de définir sa précision moyenne empirique $\hat{\mu}_k$:

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{s=1}^{t-1} a_t \mathbf{1}_{I_s=k} \quad \text{avec} \quad N_k = \sum_{s=1}^{t-1} \mathbf{1}_{I_s=k} \,. \tag{1}$$

L'entier N_k représente le nombre de fois où le bras k a été choisi avant le tour t. La moyenne μ_k est ensuite utilisée par Mutant-UCB pour choisir de manière optimiste le bras suivant à évaluer, en résolvant l'équation :

$$I_t \in \underset{k\in\{1,\dots K\}}{\operatorname{argmax}} \left\{ \hat{\mu}_k + \sqrt{\frac{E}{N_k}} \right\}. \tag{2}$$

Avec E un paramètre d'exploration à fixer. Nous introduisons à présent l'entier $\overline{N}_k$ qui compte le nombre de fois où le modèle associé au bras k a été entraîné. Nous posons N la valeur maximale que peuvent prendre les N_k. Une fois le bras I_t choisi, en prenant $p_t = 1 - \overline{N}_{I_t}/N$, Mutant-UCB :

- effectue un sous-entraînement sur $f_{I_t}^{\overline{N}_{I_t}}$ avec une probabilité p_t ou
- utilise l'opérateur de mutation sur $f_{I_t}^{\overline{N}_{I_t}}$ avec une probabilité $1 - p_t$.

La mutation est effectuée sur le modèle entraîné $f_{I_t}^{\overline{N}_{I_t}}$ et non uniquement sur f_{I_t} afin d'inclure le cas où certains paramètres du modèle optimisés pendant l'entraînement (les poids d'un réseau de neurones pax exemple) seraient transmis à son mutant. Nous détaillons l'opération de mutation pour les expériences dans la Section 4. La mutation créée un nouveau modèle qui est immédiatement entraîné et ajouté à la liste des modèles potentiels à conserver à la fin de l'algorithme. Ainsi, le nombre de modèles K augmente de 1 à chaque fois qu'un modèle mutant est créé. La probabilité p_t, elle, diminue au fur et à mesure que le modèle choisi a été entraîné. Il devient ainsi plus susceptible de muter. La probabilité limite le nombre de sous-entraînements d'un même modèle à N. L'idée sous-jacente est que multiplier les sous-entraînements sera inutile voir nuisible dans le cas des réseaux de neurones qui peuvent sur-apprendre. Si l'algorithme sélectionne ce modèle déjà bien entraîné, c'est parce qu'il est probablement performant et il faut donc chercher dans ses mutants pour trouver mieux. Notons que la probabilité p_t est linéaire en N_{I_t} ; ce choix est arbitraire et nous aurions très bien pu choisir un autre type de relation, par exemple, $p_t = 1 - \exp(N_{I_t} - N)$. Finalement, l'algorithme se termine par une dernière phase durant laquelle le meilleur modèle trouvé durant les T itérations, $\widehat{I}_T$, est sélectionné et son entraînement est finalisé par $N - \overline{N}_{\widehat{I}_T}$ sous-entraînements supplémentaires.

Remarque 3.1. L'utilisation de l'opérateur de mutation repose sur l'intuition que la distribution des performances des mutants d'un certain modèle j est différente de la distribution générale des performances dans notre espace de recherche, et une hypothèse sous-jacente pourrait être par exemple :

$$\mathbb{E}\Big[\, \mu_k \,\Big|\, f_k \text{ est un mutant de } f_j^{\overline{N}_j} \,\Big] = \mu_j \,.$$

4 Expériences

Dans cette section, nous évaluons les performances de Mutant-UCB, sur l'optimisation des réseaux de neurones. Afin de mettre en évidence les avantages de notre méthode, nous nous plaçons dans un cas où nous ne faisons aucune hypothèse sur la régularité de la récompense *acc* et où nous ne considérons aucune distance entre les éléments f_k de notre espace de recherche.

Algorithm 1 Mutant-UCB

Paramètres :
Budget T, paramètre d'exploration E, nombre de modèles K, nombre de sous-entraînements maximal pour un modèle N

Initialisation
Choisir aléatoirement K modèles non-entraînés $f_1, \dots, f_K$
Pour $k = 1, 2, \dots, K$
Faire un premier sous-entraînement pour f_k, qui devient f_k^1 et récupérer $a_k = \mathrm{acc}\big(f_k^1, \mathcal{D}_{\text{valid}}\big)$
Définir $N_k = \overline{N}_k = 1$, $\hat{\mu}_k = a_k$

Pour $t = K+1, K+2, \dots, (T-N+1)$
Prendre $I_t \in \mathrm{argmax}_{k\in\{1,\dots K\}} \left\{\hat{\mu}_k + \sqrt{\frac{E}{N_k}}\right\}$ et tirer $X_t \sim \mathcal{B}(p_t)$ avec $p_t = 1 - \overline{N}_{I_t}/N$
Si $X_t = 1$:
Faire un sous-entraînement : $f_{I_t}^{\overline{N}_{I_t}}$ devient $f_{I_t}^{\overline{N}_{I_t}+1}$ et récupérer $a_t = \mathrm{acc}\big(f_{I_t}^{\overline{N}_{I_t}+1}, \mathcal{D}_{\text{valid}}\big)$
Mettre à jour $\hat{\mu}_{I_t} = \frac{1}{\overline{N}_{I_t}+1}\big(a_t + \overline{N}_{I_t}\hat{\mu}_{I_t}\big)$, $N_{I_t} = N_{I_t} + 1$ et $\overline{N}_{I_t} = \overline{N}_{I_t} + 1$
Sinon :
Mettre à jour $K = K + 1$ et créer f_K depuis $f_{I_t}^{\overline{N}_{I_t}}$
Faire un premier sous-entraînement pour f_K qui devient f_K^1 récupérer $a_t = \mathrm{acc}\big(f_K^1, \mathcal{D}_{\text{valid}}\big)$
Définir $N_K = \overline{N}_K = 1$, $\hat{\mu}_K = a_t$ et mettre à jour $N_{I_t} = N_{I_t} + 1$

Finalisation
Sélectionner le meilleur modèle $\widehat{I}_T \in \mathrm{argmax}_{k\in\{1,\dots K\}}\, \hat{\mu}_k$ et effectuer $N - \overline{N}_{\widehat{I}_T}$ sous-entraînements

Sortie : $f_{\widehat{I}_T}^N$

Nous comparons donc nos méthodes à trois algorithmes applicables dans ce cas : une recherche aléatoire, l'algorithme Hyperband et un algorithme évolutionnaire. Cette optimisation des réseaux de neurones est appliquée à trois jeux de données de classification d'images.

4.1 Présentation des expériences

Jeux de données. Nous avons réalisé nos expériences à l'aide de trois jeux de données de classification d'images, utilisés par Li et al. 2018 pour présenter l'algorithme Hyperband : CIFAR-10 (Krizhevsky et al., 2009), Street View House Numbers (SVHN, voir Netzer et al., 2011) et une version d'MNIST tournées et avec des images d'arrière-plan, appelée MRBI (Larochelle et al., 2007). CIFAR-10 et SVHN contiennent des images RVB de 32×32, des images 28×28 en niveaux de gris pour MRBI. Les labels sont convertis en entiers entre 0 et 9. Chaque jeu de données est divisé en trois : le jeu d'entraînement, de validation et de test. Le jeu d'entraînement est utilisé pour optimiser les poids du modèle (c'est-à-dire pour effectuer les sous-entraînements) et celui de validation pour obtenir les précisions. Enfin, la précision des modèles finaux de chacun des algorithmes est calculée sur le jeu de test. CIFAR-10 comporte 35k images dans le jeu d'entraînement, 15k dans celui de validation et 10k dans celui de test, SVHN 51k, 22k et 26k et MRBI 10k, 2k et 50k. Pour tous les jeux de données, nous avons normalisé les images de manière à ce que les images aient une moyenne de zéro et un écart-type de un.

L'espace de recherche Nous avons utilisé le *framework DRAGON* développé par Keisler et al. 2023 pour représenter nos réseaux de neurones. L'article contient une explication et

un tutoriel pour utiliser le package associé. Dans ce *framework*, les réseaux de neurones sont représentés par des graphes acycliques dirigés (DAGs), où les nœuds représentent les couches (par exemple récurrentes, *feed-forward*, convolutives) et les arêtes représentent les connexions entre elles. La tâche sur laquelle nous voulons tester nos algorithmes est la classification d'images. Pour ce faire, nous définissons un espace de recherche (l'ensemble des modèles possibles f_k) dédié à cette tâche. Tout modèle échantillonné f_k est ainsi composé de DAGs. Le premier traite des données 2D et peut être constitué de convolutions 2D, de *pooling* 2D, de normalisation et de *dropout*. Le second est constitué d'une couche de *flatten* suivie de couches MLP (*Multi-Layers Perceptrons*) et de normalisation. Une dernière couche MLP est ajoutée à la sortie du modèle pour convertir le vecteur latent dans le format de sortie souhaité. Le *framework* propose un certain nombre d'opérateurs, pour optimiser ces graphes, tels que des mutations et des *crossover*. Les opérateurs de mutation modifient l'architecture du réseau de neurones en ajoutant, supprimant ou modifiant les nœuds et les connexions dans le graphe. Ils peuvent également être appliqués à l'intérieur des nœuds, sur les hyperparamètres du réseau de neurones (par exemple, la taille du noyau de la couche de convolution ou une fonction d'activation). Le *crossover* consiste à échanger les parties de deux graphes.

Sous-entraînements. Nous avons entraîné nos réseaux de neurones à l'aide d'un taux d'apprentissage cyclique comme suggéré par Huang et al. 2017. Quand le taux d'apprentissage est faible, le réseau de neurones atteint un minimum local et lorsqu'il ré-augmente, le modèle en sort. Dans nos expériences, un sous-entraînement correspond à l'une de ces boucles, avec un taux d'apprentissage passant de son maximum à son minimum. Nous notons N le nombre maximum de sous-entraînements par élément f_k de notre espace de recherche.

Baselines. La recherche aléatoire (RS), l'algorithme évolutionnaire (EA), Hyperband et Mutant-UCB ont tous été implémentés de manière à pouvoir être utilisés avec DRAGON. Ils utilisent les mêmes fonctions de d'entraînement et de validation pour évaluer les performances des réseaux de neurones et ont le même nombre de ressources T. Pour la recherche aléatoire, nous sélectionnons au hasard $K_{\mathrm{RF}} = T/N$ réseaux de neurones. Pour chacun d'entre eux, nous effectuons N sous-entraînements. Pour l'algorithme évolutionnaire, nous avons implémenté une version asynchrone. Par rapport à l'algorithme standard, la version asynchrone est plus efficace dans un environnement HPC (*High Performance Computing*) car elle crée deux nouveaux individus dès qu'un processus est disponible, et n'attend pas que toute la population soit évaluée (Liu et al., 2018). Nous définissons une population initiale de taille K_{EA}, de réseaux de neurones initiaux et nous les entraînons N fois. Ensuite, nous faisons évoluer la population à l'aide des opérateurs de mutation et de *crossover* de DRAGON et générons au total $T/N - K_{\mathrm{EA}}$ descendants, tous entraînés N fois. Si un descendant généré est meilleur que le plus mauvais modèle présent dans la population, il le remplace. Pour Hyperband, nous avons exécuté l'algorithme avec ses paramètres R et η de sorte que le nombre total de sous-entraînements soit T et que chaque modèle ne puisse être entraîné plus de N fois (voir Li et al., 2018 pour plus de détails). L'algorithme Mutant-UCB démarre avec une population initiale de $K_{\textsc{Mutant}}$ et utilise un budget de T. Pour une comparaison équitable, EA et Mutant-UCB utilisent les mêmes opérateurs de mutation. Nous fixons $K_{\mathrm{EA}} << K_{\textsc{Mutant}} \lesssim K_{\mathrm{RF}}$. En effet, chaque modèle est entièrement entraîné par l'algorithme évolutionnaire, K_{EA} doit être plus faible que T/N pour permettre la génération d'un nombre suffisant de descendants. Avec

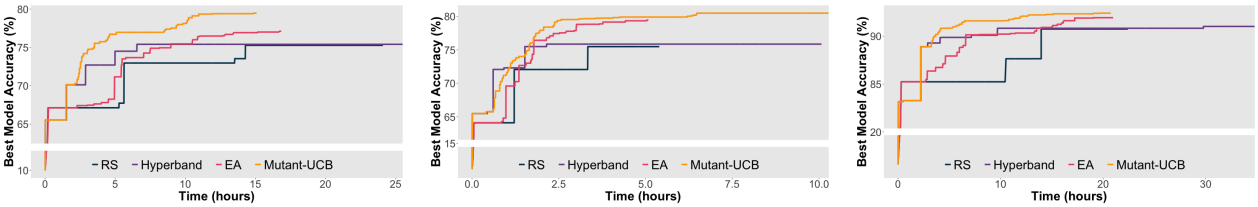
l'allocation de ressources, Mutant-UCB pourra générer et évaluer plus que T/N modèle (c'est aussi le cas pour HyperBand), d'où $K_{\text{MUTANT}} >> K_{\text{EA}}$.

4.2 Résultats

TABLE 1 – Nombre de modèles testés et précision (en %) du meilleur modèle pour la recherche aléatoire (RS), l'algorithme évolutionnaire (EA) et Mutant-UCB sur CIFAR-10, MRBI et SVHN.

Jeu de données	CIFAR-10	MRBI	SVHN
RS	1 000 · 75.3	1 000 · 75.5	1 000 · 90.7
EA	1 000 · 77.1	1 000 · 79.5	1 000 · 91.9
Hyperband	2 400 · 75.4	2 400 · 75.9	2 400 · 91.0
Mutant-UCB	**3 399 · 79.5**	**3 463 · 80.5**	**3 471 · 92.4**

Pour nos expériences, nous fixons $T = 10\,000$, $N = 10$ et $E = 0,05$ pour Mutant-UCB. Chaque sous-entraînement contient 10 époques, ce qui donne un maximum de 100 époques d'entraînement, et le taux d'apprentissage est fixé à 0,01. Chaque expérience est réalisée dans un environnement HPC avec de 20 GPUs NVIDIA V100. La Table 1 détaille les précisions (*accuracies*) maximales et le nombre de modèles testés pour chaque algorithme. Mutant-UCB bat les autres algorithmes pour tous les jeux de données. L'utilisation de la mutation semble être le facteur principal de cette performance puisque l'algorithme évolutionnaire devance largement Hyperband et la recherche aléatoire. Cependant, l'allocation de ressources joue également un rôle car Hyperband est meilleur que la recherche aléatoire et mutant meilleur que l'algorithme évolutionnaire, ce algorithmes convergent plus rapidement, comme on peut le constater sur la Figure 1. Les temps de calcul pour effectuer les T itérations varient beaucoup entre les algorithmes et les jeux de données, d'abord pour des raisons matérielles, mais aussi à cause de l'allocation de ressources. Tout au long de l'article, nous supposons implicitement que le budget d'un sous-entraînement en terme de stockage et de temps de calcul est indépendant du modèle, ce qui est inexact. Un entraînement plus long de modèles complexes peut prendre plus de temps et affecte la durée totale de l'expérience. Mutant-UCB, avec l'opérateur de mutation et l'allocation des ressources, a la convergence la plus rapide et donne les meilleures précisions.



(a) CIFAR-10. (b) MRBI. (c) SVHN.

FIGURE 1 – Précision (*accuracy*) du meilleur modèle en fonction du temps de calcul pour la recherche aléatoire (RS), l'algorithme évolutionnaire (EA) et Mutant-UCB sur CIFAR-10, MRBI et SVHN.

5 Conclusion

Mutant-UCB, un algorithme de sélection de modèles innovant, qui combine un algorithme de bandit de type UCB avec un opérateur de mutation originaire des algorithmes évolutionnaires. La plupart des approches de sélection de modèless, telles que l'optimisation bayésienne ou les algorithmes de bandits continus, considèrent un espace vectoriel normé pour représenter l'ensemble des configurations possibles. Ces approches supposent que la fonction de récompense est suffisamment régulière sur cet espace normé, signifiant que deux configurations proches dans cet espace sous-jacent ont des précisions proches. Mutant-UCB et les autres algorithmes de nos expériences ne nécessitent pas ce type d'hypothèse. En outre, grâce à l'allocation des ressources, Mutant-UCB présente un potentiel exploratoire élevé. Il peut évaluer plus de modèles avec un même budget que la recherche aléatoire ou les algorithmes évolutionnaires. Par exemple, sur le jeu de données MRBI, avec un budget $T = 10\ 000$, Mutant-UCB a évalué 3 500 modèles, alors que l'algorithme évolutionnaire et la recherche aléatoire n'en ont évalué que 1000. L'utilisation de la mutation, qui peut être vue comme un concept de proximité sans nécessiter d'espace normé, renforce l'exploitation de solutions prometteuses et nous permet d'atteindre des modèles beaucoup plus performants que ceux trouvés par Hyperband et la recherche aléatoire. Elle reste plus permissive que le *crossover* qui nécessite l'homogénéité entre les éléments de l'espace de recherche. Ainsi, Mutant-UCB pourrait être utilisé avec un espace de recherche combinant différents modèles de machine learning, tels que les réseaux de neurones, les forêts aléatoires ou le *boosting*, à condition de définir un opérateur de mutation pour chaque type de modèle. Enfin, notre algorithme peut être implémenté de manière efficace sur HPC car les modèles sont évalués de manière indépendante et asynchrone, contrairement à Hyperband et aux algorithmes évolutionnaires classiques, qui évaluent les populations de manière synchrone. En résumé, l'algorithme Mutant-UCB présente plusieurs avantages qui en font un algorithme attrayant, en plus de ses performances démontrées dans la section précédente. La flexibilité de cet algorithme signifie qu'il peut être appliqué à un large éventail de problèmes. Une extension naturelle de cet article serait d'appliquer Mutant-UCB à une variété de tâches, de modèles et d'espaces de recherche où les algorithmes à l'état de l'art seraient limités voir inutilisables.

Références

Audibert, J.-Y., S. Bubeck, and R. Munos (2010). Best arm identification in multi-armed bandits. In *COLT*, pp. 41–53.

Auer, P., N. Cesa-Bianchi, and P. Fischer (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning 47*, 235–256.

Garrido-Merchán, E. C. and D. Hernández-Lobato (2020). Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing 380*, 20–35.

Huang, G., Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger (2017). Snapshot ensembles : Train 1, get m for free. *arXiv preprint arXiv :1704.00109*.

Hutter, F., L. Kotthoff, and J. Vanschoren (2019). *Automated machine learning : methods, systems, challenges*. Springer Nature.

Kandasamy, K., G. Dasarathy, J. B. Oliva, J. Schneider, and B. Póczos (2016). Gaussian process bandit optimisation with multi-fidelity evaluations. *Advances in neural information processing systems 29*.

Keisler, J., E.-G. Talbi, S. Claudel, and G. Cabriel (2023). An algorithmic framework for the optimization of deep neural networks architectures and hyperparameters. *arXiv preprint arXiv :2303.12797*.

Krizhevsky, A., G. Hinton, et al. (2009). Learning multiple layers of features from tiny images.

Larochelle, H., D. Erhan, A. Courville, J. Bergstra, and Y. Bengio (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pp. 473–480.

Lattimore, T. and C. Szepesvári (2020). *Bandit algorithms*. Cambridge University Press.

Li, K., A. Fialho, S. Kwong, and Q. Zhang (2013). Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation 18*(1), 114–130.

Li, L., K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar (2018). Hyperband : A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research 18*(185), 1–52.

Liu, H., K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu (2018). Hierarchical representations for efficient architecture search.

Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng (2011). Reading digits in natural images with unsupervised feature learning.

Srinivas, N., A. Krause, S. M. Kakade, and M. Seeger (2009). Gaussian process optimization in the bandit setting : No regret and experimental design. *arXiv preprint arXiv :0912.3995*.

Strumberger, I., E. Tuba, N. Bacanin, R. Jovanovic, and M. Tuba (2019). Convolutional neural network architecture design by the tree growth algorithm framework. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE.

Valko, M., N. Korda, R. Munos, I. Flaounas, and N. Cristianini (2013). Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv :1309.6869*.

Young, S. R., D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton (2015). Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the workshop on machine learning in high-performance computing environments*, pp. 1–5.

Zoph, B. and Q. Le (2016). Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*.