TabM: Advancing Tabular Deep Learning with Parameter-Efficient Ensembling

Faten Racha Said, Raphaël Renard, Luc Salvon

Points clés

- Réimplémentation de TabM et ses variantes.
- Proposition de nouvelles architectures pour approfondir l'aspect ensembling de l'article.
- Analyses (métriques, diversité des sous-modèles...).
- 13 modèles codés, 5 datasets exploités, 9 métriques

Introduction

Le traitement des données tabulaires a longtemps été dominé par les méthodes GBDT, reconnues pour leur efficacité sur ce type de données. Cependant, ces approches présentent des limitations lorsqu'il s'agit de capturer des **représentations réutilisables** ou de s'adapter à diverses tâches. TabM propose une nouvelle perspective : tirer parti d'un simple réseau type MLP pour imiter un ensemble de modèles tout en conservant une **structure légère**. En combinant simplicité et performances comparables à l'état de l'art, TabM ouvre la voie à de **nouvelles** expérimentations et architectures prometteuses.

TabM et ses variantes

Contrairement à un ensemble classique où chaque modèle dispose de matrices de poids indépendantes, nous utilisons **BatchEnsemble** (Backbone) qui factorise les poids en combinant une matrice partagée W avec des matrices r_i et s_i spécifiques à chaque sous-modèle (*adaptateurs*).

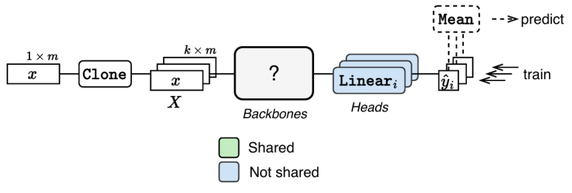


Figure 1:Architecture commune des différentes variantes de TabM.

- **TabM$_{\text{naive}}$** : Applique BatchEnsemble à toutes les couches linéaires d'un MLP standard, avec des têtes de prédiction non partagées.
- **TabM$_{\text{mini}}$** : Supprime tous les adaptateurs spécifiques à chaque sous-modèle sauf le premier (R).
- **TabM** : Tous les adaptateurs sont gardés mais initialisés à 1, sauf le premier.
- **TabM$_{\text{mini}}^{\dagger}$ & TabM†** : Variantes utilisant des embeddings non linéaires (piecewise-linear embeddings).

Expérimentations et diversité

Analyser la diversité des sorties des sous-modèles avec :

- **Corrélation de Spearman** moyenne entre les prédictions des sous-modèles.
- **Divergence KL** entre les distributions des prédictions de chaque sous-modèle et la distribution moyenne.
- **Visualisation des distributions** des sous-modèles à chaque couche à l'aide d'une t-SNE.
- **Différentes initialisations** de R et S (uniforme, normale, laplacienne) avec des facteurs multiplicatifs.
- Variante architecturale avec une transformation **non linéaire** (via une fonction d'activation) appliquée à R et S à chaque couche, visant à augmenter la diversité et l'expressivité des sous-modèles.

Agrégation via l'incertitude

Pour la **classification**, nous proposons une méthode d'**agrégation basée sur la confiance** des modèles, calculée à partir de l'**incertitude** des résultats.

On établit la mesure de confiance $C(\hat{y}_i) = \frac{H_{max} - H(\hat{y}_i)}{H_{max}}$, avec H l'entropie de Shannon, H_{max} l'entropie maximale pour la dimension de sortie.
La mesure de confiance est alors bornée par 0 (probabilités de classes équivalentes) et 1 (une classe a une probabilité de 1).
Pour agréger les k modèles, on pondère alors chacune des prédictions par sa confiance, en appliquant un SoftMax sur les confiances pour garantir une bonne agrégation.
Cette méthode est **applicable à tous les modèles ensemblistes**, notée par exemple TabM$_{conf}$.

Note : On remarque une surconfiance des modèles sur des données OOD, que nous n'avons pas su aligner sur l'accuracy. Aligner la bonne prédiction et la confiance permettrait sûrement d'obtenir de meilleurs résultats.

Pruning

Objectif : Réduire les coûts computationnels (mémoire et temps) via des stratégies de **pruning adaptatif**. Bien que l'article propose une méthode de pruning, celle-ci est moins performante que TabM.

Nous proposons donc une **stratégie alternative** :
Pour chaque epoch :

1. Entraîner le modèle.
2. Évaluer le modèle avec plusieurs **keep ratios** (100 %, 75 %, 50 %, 25 % des sous-modèles).
3. Mettre à jour l'architecture en conservant le ratio qui minimise la *loss*.

Mécanisme d'Attention

Objectif : Intégrer un mécanisme d'attention pour pondérer dynamiquement l'importance des colonnes. Plutôt que traiter chaque colonne de manière équivalente, l'idée est de moduler leurs contributions en fonction de chaque instance pour **capturer les dépendances contextuelles** complexes entre elles.

Méthode [Fig 2]:

1. Pour chaque instance x, générer son vecteur d'*embedding* et le traiter à travers des **blocs d'attention** dont on fait la moyenne afin d'obtenir une seule représentation contextualisée de x.
2. L'instance contextualisée est transmise à **BatchEnsemble**, qui capture les relations globales à l'aide de W, tout en ajoutant les perturbations (r_i, s_i) pour capturer les variations fines pour chaque membre de l'ensemble. La prédiction finale est produite à l'image de TabM.

Pour une meilleure interprétabilité, nous sauvegardons les poids d'attention afin de **visualiser** les caractéristiques globalement importantes.

Architecture

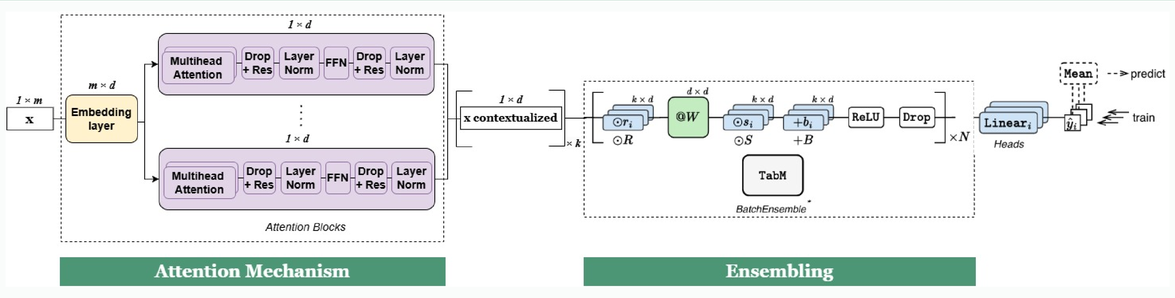


Figure 2:Notre architecture de TabM avec mécanisme d'attention.

Résultats

- **Diversité des sous-modèles**

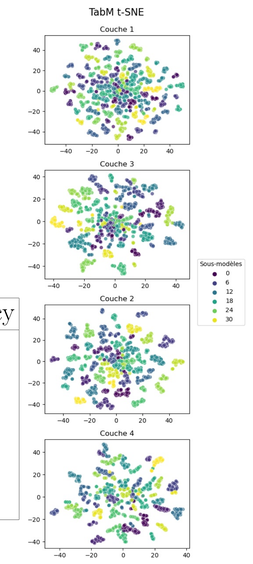
Modèle	Spearman	KL div
TabM	0.0628	0.6180
TabM$_{naive}$	0.1202	0.2539
TabM$_{mini}$	0.1160	0.2724
MLP$_k$	0.1050	0.5854
TabM†	0.1569	0.0829
TabM$_{NonLinéaire}$	0.1283	0.0689

Table 1:Comparaison des modèles sur Wine selon Spearman et KL divergence.

Scale	Distribution	Corrélation	Accuracy
0.5	Uniform	0.145	0.354
0.5	Normal	0.131	0.361
0.5	Laplace	0.154	0.354
2.0	Uniform	0.180	0.376
2.0	Normal	0.280	0.391
2.0	Laplace	0.228	0.341

Table 2:Comparaison des performances sur Wine selon l'échelle et la distribution.

- **Prédictions des sous-modèles (Wine)**



- **Comparaison des performances des modèles**[a]

Modèle	Accuracy	Loss	F1-Score	Temps(s)
MLP	0.33	-	0.98	1.75
XGBoost	0.93	-	0.93	**0.04**
FT-Transformer	**0.97**	0.18	**0.97**	7.52
Excel-Former	0.93	0.14	0.93	9.12
T2G-Former	**0.97**	0.19	**0.97**	8.82
TabM$_{\text{naive}}$	0.77	0.98	0.73	6.81
TabM$_{\text{mini}}$	0.67	0.96	0.54	5.30
TabM	0.83	0.42	0.83	6.53
TabM$_{\text{conf}}$	0.87	0.28	0.87	6.66
TabM$_{\text{Attention}}$	**0.97**	0.26	**0.97**	5.89
TabM$_{\text{pruned9}}$	0.33	**0.04**	0.17	7.66
TabM$_{\text{NonLinéaire}}$	0.87	0.79	0.87	5.07
TabM†	0.90	0.54	0.90	7.87

Table 3:Comparaison des modèles pour le dataset iris.

Les méthodes basées sur les arbres offrent le meilleur **compromis** entre performance et rapidité. Les modèles neuronaux et variantes TabM, notamment notre variante avec Attention, sont **compétitifs** en classification mais moins efficaces en régression, tandis que les versions basées sur la **confiance** apportent un léger bénéfice. Les résultats du **pruning** sont parfois instables mais montrent souvent les meilleures *loss*. A noter que TabM†, bien que donnant de bons résultats, augmente la taille des données et est donc **très lent** sur de grands datasets. Les modèles transformers de **l'état de l'art** ont des performances similaires à TabM$_{\text{Attention}}$ en ayant une complexité plus élevée.

[a]https://github.com/said-racha/Expanding-the-Power-of-Tabular-Deep-Learning