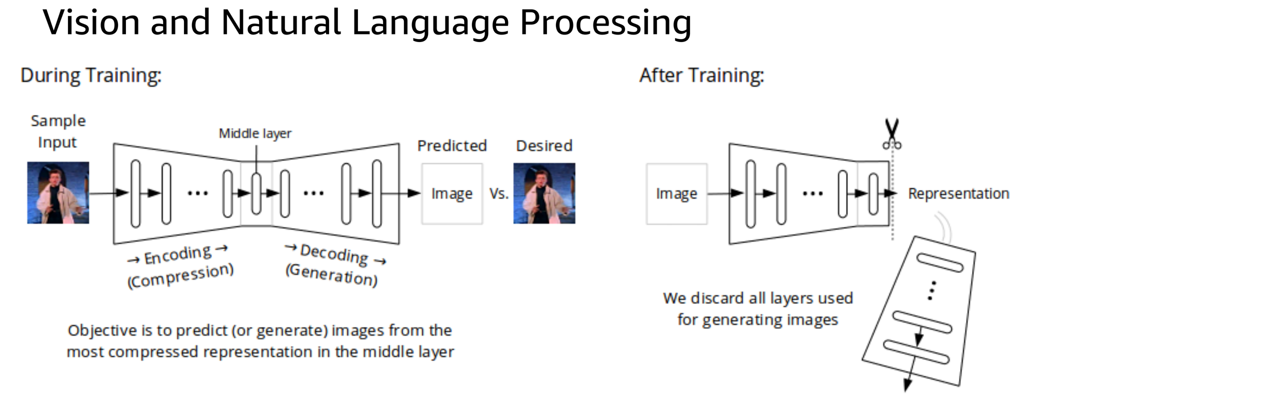
amazon | science
Buyer Risk Prevention

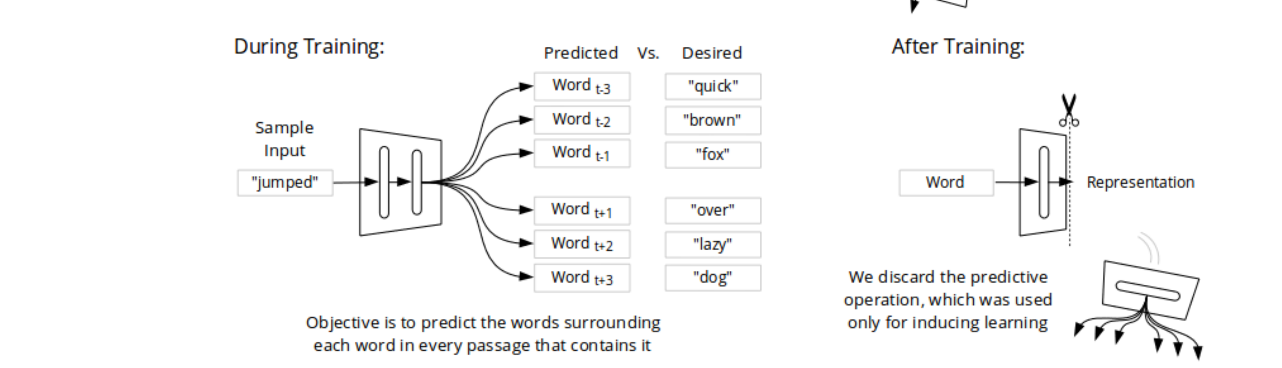
SwitchTab: Switched Autoencoders are Effective Tabular Learners

Jing Wu*, Suiyao Chen*, Qi Zhao,
Renat Sergazinov, Chen Li, Shengjie Liu, Chongchao Zhao, Tianpei Xie, Hanqing Guo, Cheng Ji, Daniel Cociorva, Hakan Brunzell
{jingwua, suiyaoc, qqzhao, renserg, chenlii, zycjlsj, zchongch, lukexie, hanqiguo, cjiamzn, cociorva, brunzell}@amazon.com
*These authors contributed equally

Background

- Representation Learning is to learn to transform data from its original representation to a new representation that retains information essential to objects that are of interest to our tasks
- Representation Learning has made remarkable advancements in Computer Vision and Natural Language Processing





Feature Decoupling

Feature Decoupling For Domain Adaption

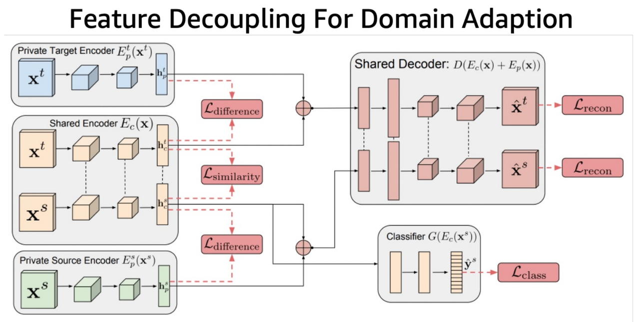


Figure 1: A shared-weight encoder $E_c(\mathbf{x})$ learns to capture representation components for a given input sample that are shared among domains. A private encoder $E_p(\mathbf{x})$ (one for each domain) learns to capture domain-specific components of the representation. A shared decoder learns to reconstruct the input sample by using both the private and source representations. The private and shared representation components are pushed apart with soft subspace orthogonality constraints $\mathcal{L}_{difference}$, whereas the shared representation components are kept similar with a similarity loss $\mathcal{L}_{similarity}$.

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., & Erhan, D. (2016). Domain separation networks. *Advances in neural information processing systems*, 29.

Feature Decoupling for Image Generation

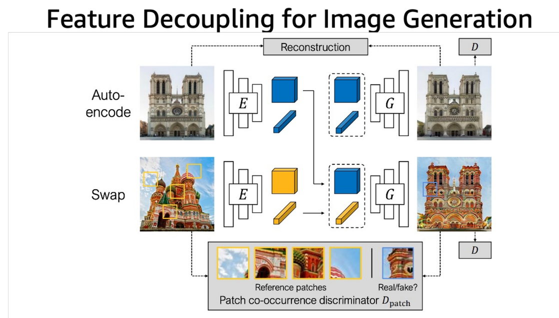


Figure 2: **Swapping Autoencoder** consists of autoencoding (*top*) and swapping (*bottom*) operation. **(Top)** An encoder E embeds an input (Notre-Dame) into two codes. The structure code is a tensor with spatial dimensions; the texture code is a 2048-dimensional vector. Decoding with generator G should produce a realistic image (enforced by discriminator D) matching the input (reconstruction loss). **(Bottom)** Decoding with the texture code from a second image (Saint Basil's Cathedral) should look realistic (via D) and match the texture of the image, by training with a patch co-occurrence discriminator D_{patch} that enforces the output and reference patches look indistinguishable.

Park, T., Zhu, J. Y., Wang, O., Lu, J., Shechtman, E., Efros, A., & Zhang, R. (2020). Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33, 7

Feature Decoupling for Salient Feature Enhancement

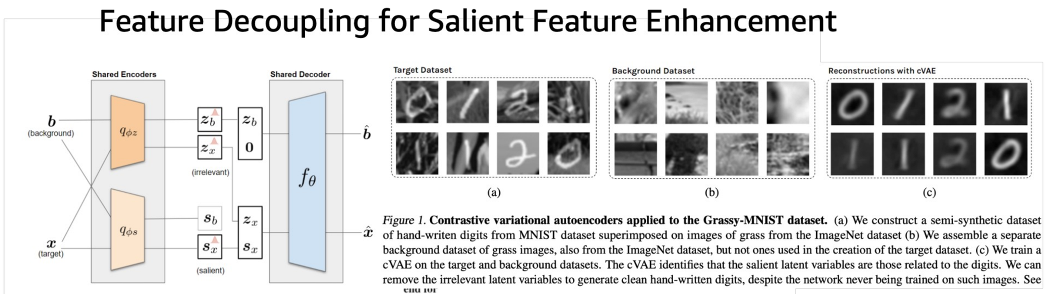


Figure 1. **Contrastive variational autoencoders applied to the Grassy-MNIST dataset.** (a) We construct a semi-synthetic dataset of hand-writen digits from MNIST dataset superimposed on images of grass from the ImageNet dataset (b) We assemble a separate background dataset of grass images, also from the ImageNet dataset, but not ones used in the creation of the target dataset. (c) We train a cVAE on the target and background datasets. The cVAE identifies that the salient latent variables are those related to the digits. We can remove the irrelevant latent variables to generate clean hand-written digits, despite the network never being trained on such images. See

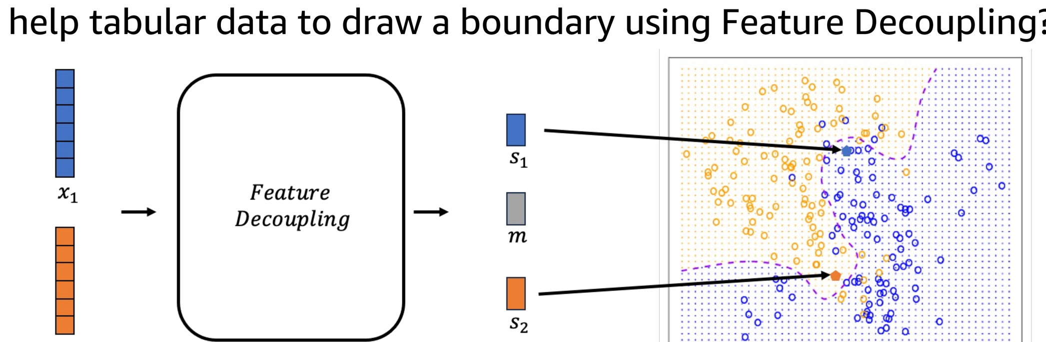
Return: trained encoders q_{ϕ_s}, q_{ϕ_z} and decoder f_θ

(a) (b)

Figure 3. **Architecture and algorithm for the cVAE.** (a) Here, we illustrate the architecture of the contrastive VAE. The background target datasets are fed into two sets of shared encoders q_{ϕ_s} and q_{ϕ_z}, which output the salient and irrelevant latent variables respectively. The latent variables for the target dataset are concatenated and fed into a decoder network, while only the (zero-padded) irrelevant variable is fed into the decoder network. The decoder outputs reconstructions of the target and background data. (b) The pseudocode for the cVAE is provided here, using batch gradient descent for simplicity.

Abid, A., & Zou, J. (2019). Contrastive variational autoencoder enhances salient features. *arXiv preprint arXiv:1902.04601*.

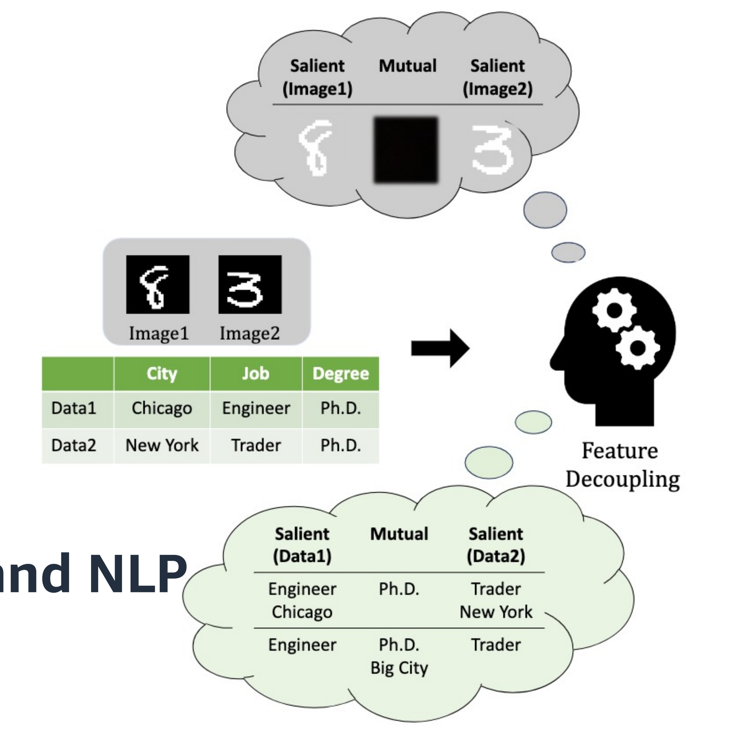
Can we help tabular data to draw a boundary using Feature Decoupling?



When comparing two data samples,
- mutual features consist of information that highlights common characteristics.
- salient features emphasize the distinctive attributes to differentiate one sample from the other.

Motivation

- Tabular Representation Learning has not been fully explored, due to
  - Inherent heterogeneity which lacks explicit spatial relationships found in images (e.g., similar background and distinct characters) or the semantic dependencies observed in languages
  - Various discrete and continuous distributions from both numerical and categorical features
  - Complex interrelationship from features that can be dependent or independent from each other



Can we adopt the success of Representation Learning from CV and NLP domains to Tabular data?

- For an image, a person can easily distinguish the salient digits from the mutual background
- Separating the salient and mutual information becomes challenging for tabular samples
- Explicitly distinguish between mutual information in global and salient information within the feature space can be useful for representation learning

Figure 1: Given a pair of images, a person can easily distinguish the salient digits and mutual background due to the well-structured spatial relationships. However, it becomes challenging to distinguish a pair of tabular samples. For instance, feature City may be salient between data points "Chicago" and "New York" for word counts, however, still sharing some latent mutual information (e.g., big cities), making it challenging for decoupling. Note that this decoupling process is for illustration only. In the implementation, all the decoupled samples are computed in the feature space.

SwitchTab Self-supervised Learning Framework

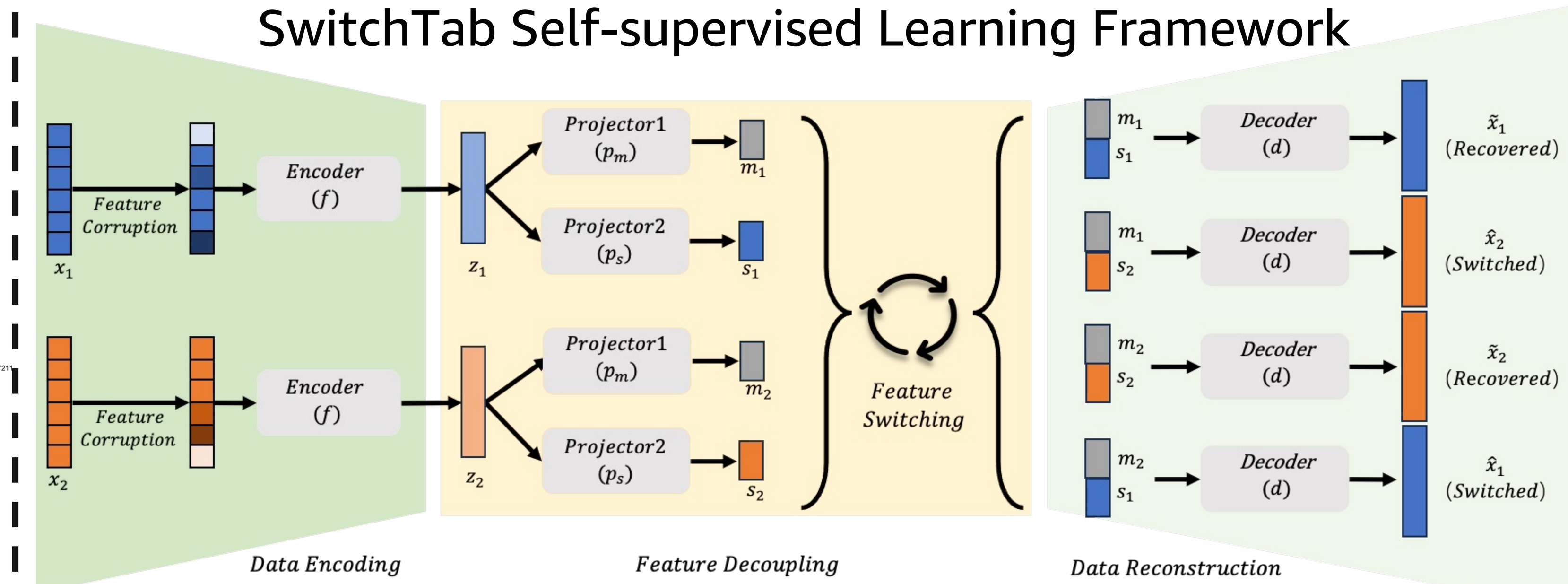
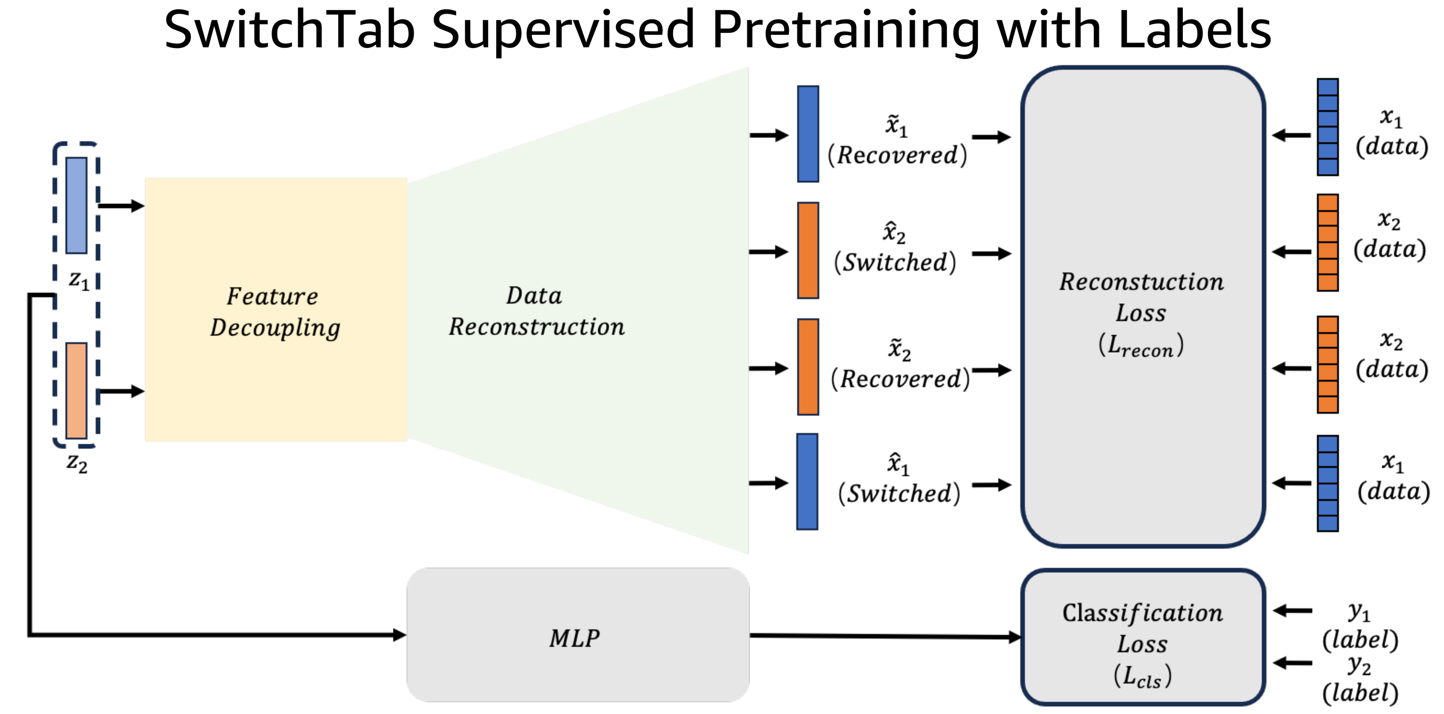


Figure 2: Block diagram of the proposed self-supervised learning framework. (1) Two different samples x_1 and x_2 are randomly corrupted and encoded into feature vectors z_1 and z_2 through encoder f. (2) feature vectors z_1 and z_2 are decoupled into mutual and salient features by two different projectors p_m and p_s, respectively. (3) Mutual and salient features are combined and reconstructed by a decoder d where the salient feature dominates the sample type and the mutual feature provides common information that is switchable among two samples.

SwitchTab Supervised Pretraining with Labels



Loss Function and Objectives

Algorithm 1: Self-supervised Learning with SwitchTab

Require: unlabeled data $\mathcal{X} \subseteq \mathbb{R}^M$, batch size B, encoder f, projector for mutual information p_m, projector for salient information p_s, decoder d, mean squared error MSE, feature concatenation $\oplus$.

for two sampled mini-batch $\{x_i^1\}_{i=1}^B \subseteq \mathcal{X}$ and $\{x_i^2\}_{i=1}^B \subseteq \mathcal{X}$ **do**
  for each sample x_i^1 and x_i^2, apply feature corruption, define the corrupted feature as: $\breve{x}_i^1$ and $\breve{x}_i^2$, for $i \in [B]$
  data encoding:
  $z_i^1 = f(\breve{x}_i^1)$, $z_i^2 = f(\breve{x}_i^2)$, for $i \in [B]$
  feature decoupling:
  (1) the salient and mutual information of the first batch be defined as follows: $s_i^1 = p_s(z_i^1)$ and $m_i^1 = p_m(z_i^1)$.
  (2) the salient and mutual information of the second batch be defined as follows: $s_i^2 = p_s(z_i^2)$ and $m_i^2 = p_m(z_i^2)$.
  data reconstruction:
  (1) let recovered pairs be defined as:
  $\tilde{x}_i^1 = d(m_i^1 \oplus s_i^1)$, $\tilde{x}_i^2 = d(m_i^2 \oplus s_i^2)$
  (2) let switched pairs be defined as:
  $\hat{x}_i^1 = d(m_i^2 \oplus s_i^1)$, $\hat{x}_i^2 = d(m_i^1 \oplus s_i^2)$
  define reconstruction loss $\mathcal{L}_{recon} =$ $\mathrm{MSE}(x_i^1, \tilde{x}_i^1) + \mathrm{MSE}(x_i^2, \tilde{x}_i^2) + \mathrm{MSE}(x_i^1, \hat{x}_i^1) + \mathrm{MSE}(x_i^2, \hat{x}_i^2)$
  update encoder f, projectors p_m and p_s, and decoder d to minimize $\mathcal{L}_{recon}$ using RMSProp.
end for

$$\mathcal{L}_{recon} = \underbrace{\frac{1}{M}\sum_{j=1}^{M}(x_{1_j} - \hat{x}_{1_j})^2 + \frac{1}{M}\sum_{j=1}^{M}(x_{2_j} - \hat{x}_{2_j})^2}_{\text{switched}} + \underbrace{\frac{1}{M}\sum_{j=1}^{M}(x_{1_j} - \tilde{x}_{1_j})^2 + \frac{1}{M}\sum_{j=1}^{M}(x_{2_j} - \tilde{x}_{2_j})^2}_{\text{recovered}}. \quad (1)$$

$$\mathcal{L}_{semi} = \mathcal{L}_{recon} + \alpha * \mathcal{L}_{cls}, \quad (2)$$

where α is used to balance the classification loss and reconstruction loss and set to 1 as default. To illustrate, the cross-entropy loss used for classification task can be defined as follow:

$$\mathcal{L}_{cls} = -\left(y_1 \log(\hat{y}_1) + y_2 \log(\hat{y}_2)\right), \quad (3)$$

where $\hat{y}_1$ and $\hat{y}_2$ are predicted labels computing a MLP, i.e., $\hat{y}_1 = \mathrm{MLP}(z_1)$ and $\hat{y}_2 = \mathrm{MLP}(z_2)$. For regression tasks, we replace the cross-entropy loss with rooted mean squared error (RMSE).

Case Study Experiments

- Dataset
  - A standard benchmark from (Gorishniy et al. 2021) with 11 datasets including California Housing (CA), Adult (AD), Helena (HE), Jannis (JA), Higgs (HI), ALOI (AL), Epsilon (EP), Year (YE), Covertype (CO), Yahoo (YA), Microsoft(MI)
  - Additional popular datasets from recent work with 7 datasets on classification tasks
    - Bank (BK), Blastchar (BC), Arrhythmia (AT), Arcene (AR), Shoppers (SH), Volkert (VO), MNIST (MN)

Dataset size	48842	65196	83733	98050	108000	500000	518012	20640	515345	709877	1200192
Feature size	14	27	54	28	128	2000	54	8	90	699	136
Method/Dataset	**AD ↑**	**HE ↑**	**JA ↑**	**HI ↑**	**AL ↑**	**EP ↑**	**CO ↑**	**CA ↓**	**YE ↓**	**YA ↓**	**MI ↓**
TabNet	0.850	0.378	0.723	0.719	0.954	0.8896	0.957	0.510	8.909	0.823	0.751
SNN	0.854	0.373	0.719	0.722	0.954	0.8975	0.961	0.493	8.895	0.761	0.751
AutoInt	0.859	0.372	0.721	0.725	0.945	0.8949	0.934	0.474	8.882	0.768	0.750
MLP	0.852	0.383	0.723	0.723	0.954	0.8977	0.962	0.499	8.853	0.757	0.747
DCN2	0.853	0.385	0.723	0.723	0.955	0.8977	0.965	0.484	8.890	0.757	0.749
NODE	0.858	0.359	0.726	0.726	0.918	0.8958	0.985	0.464	**8.784**	0.753	0.745
ResNet	0.854	**0.396**	0.727	0.727	**0.963**	0.8969	0.964	0.486	8.846	0.757	0.748
FT-Transormer	0.859	0.391	0.729	0.729	0.960	0.8982	0.970	0.459	8.855	0.756	0.746
XGBoost	0.874	0.377	0.724	0.728	0.924	0.8799	0.964	0.431	8.819	**0.732**	**0.742**
CatBoost	0.873	0.388	0.727	0.729	0.948	0.8893	0.950	**0.423**	8.837	0.740	0.743
SwitchTab (Self-Sup.)	0.867	0.387	0.726	0.724	0.942	0.8928	0.971	0.452	8.857	0.755	0.751
SwitchTab	**0.881**	0.389	**0.731**	**0.733**	0.951	**0.8987**	**0.989**	0.442	8.822	0.744	**0.742**

Table 1: Comparison of different methods on the previous benchmark. For each dataset, the best results are shown in **Bold**. Reported results are averaged over three trials. Notations: ↓∼ RMSE for regression task, ↑∼ accuracy for classification task.

- SwitchTab consistently achieves optimal or near-optimal performance in most of the classification task
- In regression tasks, traditional methods like XGBoost or CatBoost still dominate and achieve the best results. However, SwitchTab remains highly competitive

Dataset size	45211			7043			452			200			12330			58310			518012		
Feature size	16			20			226			783			17			147			54		
Dataset	**BK**			**BC**			**AT**			**AR**			**SH**			**VO★**			**MN★**		
Raw Feature (x)	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓
Salient Feature (s)		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓		✓	✓
Logistic Reg.	0.907	0.910	0.918	0.892	0.894	0.902	0.862	0.862	0.869	0.916	0.915	0.922	0.870	0.871	0.882	0.539	0.545	0.551	0.899	0.907	0.921
Random Forest	0.891	0.895	0.902	0.879	0.880	0.899	0.850	0.853	0.885	0.809	0.810	0.846	0.929	0.931	0.933	0.663	0.669	0.672	0.938	0.940	0.945
XGboost	0.929	0.929	0.938	0.906	0.907	0.912	0.870	0.872	0.904	0.824	0.828	0.843	0.925	0.924	0.931	0.690	0.691	0.693	0.958	0.961	0.964
LightGBM	0.939	0.939	0.942	0.910	0.910	0.915	0.887	0.889	0.903	0.821	0.826	0.831	0.932	0.933	0.944	0.679	0.682	0.686	0.952	0.955	0.963
CatBoost	0.925	0.928	0.937	0.912	0.910	0.919	0.879	0.880	0.899	0.825	0.828	0.877	0.931	0.934	0.942	0.664	0.671	0.682	0.956	0.958	0.968
MLP	0.915	0.917	0.923	0.892	0.895	0.902	0.902	0.905	0.912	0.903	0.904	0.908	0.887	0.891	0.910	0.631	0.633	0.642	0.939	0.941	0.948
VIME	0.766	-	-	0.510	-	-	0.653	-	-	0.610	-	-	0.744	-	-	0.623	-	-	0.958	-	-
TabNet	0.918	-	-	0.796	-	-	0.521	-	-	0.541	-	-	0.914	-	-	0.568	-	-	0.968	-	-
TabTransformer	0.913	-	-	0.817	-	-	0.700	-	-	0.868	-	-	0.927	-	-	0.580	-	-	0.887	-	-
SAINT	0.933	-	-	0.847	-	-	**0.941**	-	-	0.910	-	-	0.931	-	-	0.701	-	-	0.977	-	-
SwitchTab(Self-Sup.)	0.917	-	-	0.903	-	-	0.900	-	-	0.904	-	-	0.931	-	-	0.629	-	-	0.969	-	-
SwitchTab	**0.942**	-	-	**0.923**	-	-	0.928	-	-	**0.922**	-	-	**0.958**	-	-	**0.708**	-	-	**0.982**	-	-

" – " indicates the experiments are not applicable for the corresponding methods to demonstrate the benefits of plug-and-play embeddings.

Table 2: Comparison of different methods on classification task. For each method, we report three categories 1) raw features only, 2) salient features only, 3) plug and play using salient features. The best results are shown in **Bold**. Columns added with ★ are multi-class classification tasks, reporting accuracy. The other results of binary classification tasks are evaluated with AUC.

- SwitchTab has remarkable performance lift in majority of the cases
- Salient features have immense value when integrated with original data as additional features to improve traditional methods

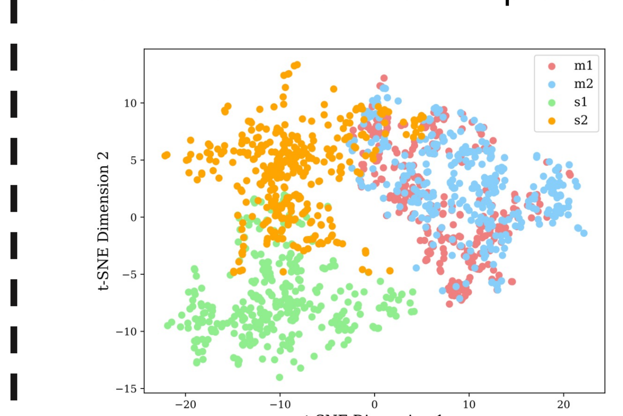


Figure 4: t-SNE visualization of mutual and salient features in two dimensional space.

- The mutual features m1 and m2 from SwitchTab, although extracted from two different classes, heavily overlap with each other.
- The salient features s1 and s2 are distinctly separated, playing a dominant role in capturing the unique properties of each class and decisively contributing to the classification boundaries.