

MultiModalPFN: Extending Prior-Data Fitted Networks for Multimodal Tabular Learning

Supplementary Material

S1. Additional Analysis

Attention Imbalance. Figure S1 illustrates the relationship between the number of non-tabular input features generated by MGM and CAP and the resulting performance across the evaluated datasets. In all cases, the best performance is achieved when the number of non-tabular features is similar to the number of tabular features. Performance tends to degrade when the number of non-tabular features is either substantially smaller or larger than the number of tabular features. These results experimentally support our analysis of attention imbalance.

Activation Choice in MGM. We study the impact of using GLU as the activation function in MGM on the CBIS-DDSM (MASS) and Salary datasets. Table S1 compares GLU against a GELU baseline in terms of accuracy and mean output-vector orthogonality. Since GLU reduces the dimensionality by half after the gating operation, the GELU baseline is configured with more parameters. Despite this advantage, GLU consistently yields higher accuracy than GELU on both datasets, while also increasing the orthogonality measure. This suggests that the gating mechanism in GLU not only improves predictive performance but also promotes more diverse output representations, aligning with our objective of learning complementary non-tabular features. Consequently, we adopt GLU as the default activation in MGM.

Table S1. **Effect of activation choice in MGM.** Comparison of accuracy and mean output-vector orthogonality when using GELU versus GLU, with and without orthogonality loss, on CBIS-DDSM (MASS) and Salary.

	CBIS-DDSM (MASS)		Salary	
Activation	Accuracy	Orthogonality	Accuracy	Orthogonality
GELU	72.09	0.0565	45.04	0.04876
GLU	75.10	0.0913	45.87	0.05831

Parameter budgets in ??. In ??, the parameter counts of the modality projector vary depending on the architectural choice. Let N denote the number of MGM heads and d the token dimension. The parameter counts scale as follows: single-head + linear $O(d)$, single-head + MLP $O(d^2 + d)$, multi-head + MLP $O(N(d^2 + d))$, and multi-head + MGM $O(N(d^2 + d/2))$. MGM requires fewer parameters than the multi-head MLP due to its linear gating with channel

splitting. These results indicate that the performance trends in ?? are not solely explained by increased model capacity. Moreover, the additional parameters introduced by the projector contribute only marginally to inference latency.

Robustness of Low-Data Regimes. In ??, MMPFN achieves a higher average performance (64.21) than TIP (58.97), despite larger relative drops on PU20 and Petfinder. This robustness primarily stems from strong priors learned during large-scale meta-training on synthetic datasets[? ?], which capture a broad range of plausible tabular distributions and enable effective generalization from few real samples. Fine-tuning then provides a light task-specific adaptation on top of this Bayesian inference. Because the model requires only light adaptation, it avoids overfitting and remains stable in low-sample settings. Together with the inductive bias of the Per-Feature Transformer, this explains the superior performance of our MMPFN across low-data experiments.

Replacing Modality Encoders. MMPFN combines pre-trained models, making the framework naturally extensible as newer and stronger encoders become available. This modular design allows components such as TabPFN or DINO to be replaced with more recent architectures without altering the overall pipeline. Leveraging improved pre-trained models enhances the quality of feature representations and can lead to measurable downstream gains. For example, substituting DINOv2 with the recently released DINOv3 yields consistent improvements across datasets, as shown in Table S2; on PU20, accuracy increases by approximately 0.74 percentage points.

We also examine the effect of replacing the text encoder. As shown in Table S3, switching between ELECTRA and DeBERTa results in only minor performance differences across the evaluated datasets. These results suggest that while stronger encoders can provide modest improvements, the overall performance of MMPFN remains relatively stable across different encoder choices.

Table S2. **Effect of replacing the image encoder.** Performance of MMPFN when substituting DINOv2 with ResNet50 and DINOv3. Results are reported as averaged accuracy over five random seeds.

Encoder	PU20	Mass	Calc	Petfinder
ResNet50	83.26	-	73.94	-
DINOv2	85.22	74.53	75.40	40.74
DINOv3	85.61	75.48	76.75	40.57

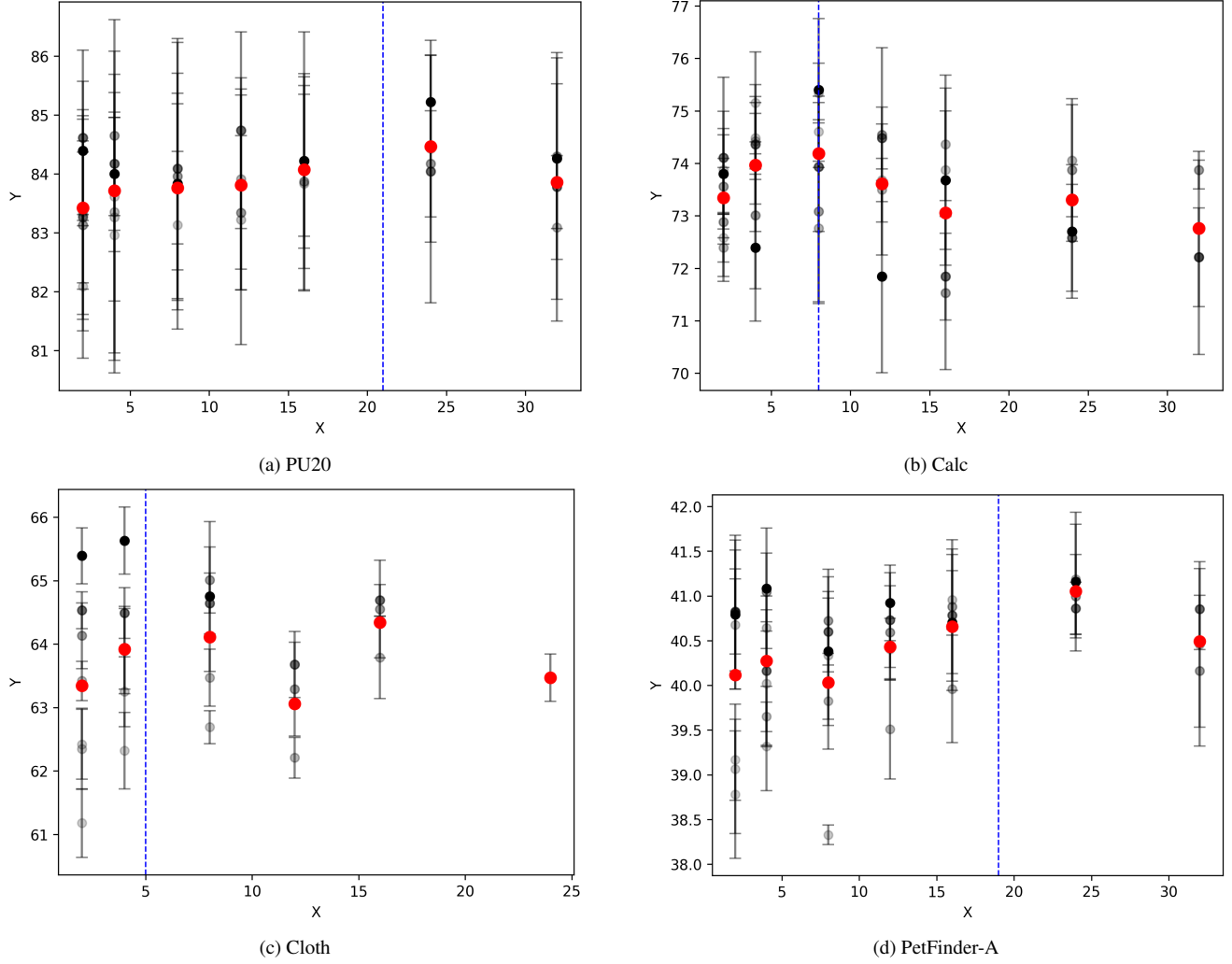


Figure S1. **Effect of the ratio between tabular and non-tabular features.** The black dots and vertical lines show the mean and variance across five random seeds. Darker black dots correspond to a larger number of MGM heads (i.e., more non-tabular features generated by MGM), ranging from 8 to 128. The red dot indicates the average result across all MGM-head settings. The x-axis shows the number of non-tabular features generated by CAP, and the y-axis denotes accuracy. The blue line represents the number of tabular features. Dataset names are shown above each subfigure.

Table S3. **Effect of replacing the text encoder.** Performance of MMPFN when using different pretrained text encoders. Results are reported as averaged accuracy over five random seeds.

Encoder	Airbnb	Salary
Electra	47.78	46.17
DeBERTa	47.82	45.69

Distribution Alignment Across Modalities. We investigate whether applying embedding-space alignment techniques—commonly used in multimodal learning—can further improve the performance of MMPFN. In particular, we incorporate Maximum Mean Discrepancy (MMD) [?], a

standard measure of distributional discrepancy [?], into our framework. We apply MMD to the embeddings generated for each feature to reduce the distributional gap between representations from different modalities. For tabular data, where feature distributions can differ substantially across dimensions, we additionally employ Joint MMD (JMMD) [?] to capture discrepancies at the level of the joint feature distribution.

We train the multi-head MLP module that produces unstructured feature embeddings by adding the discrepancy between tabular embeddings and image/text embeddings as an auxiliary loss term. However, as shown in Figure S3, this alignment-based regularization consistently underperforms

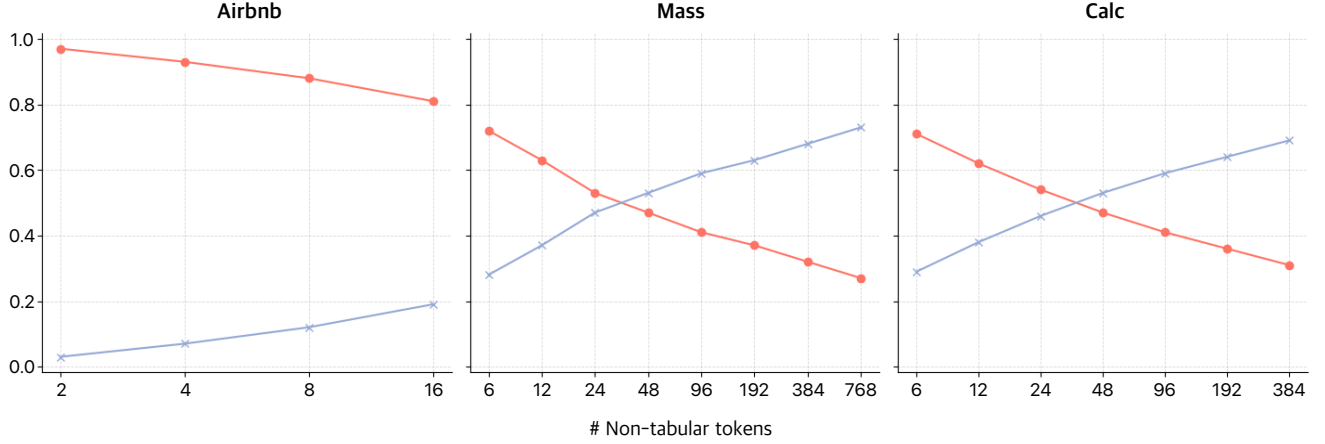
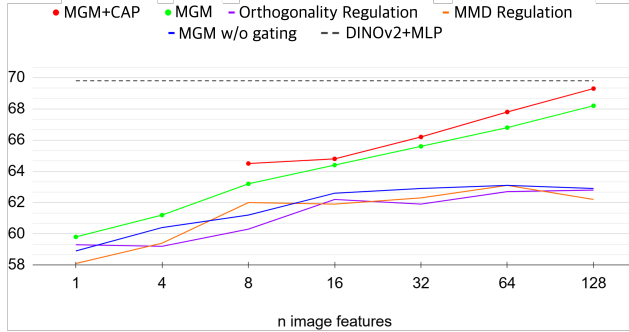
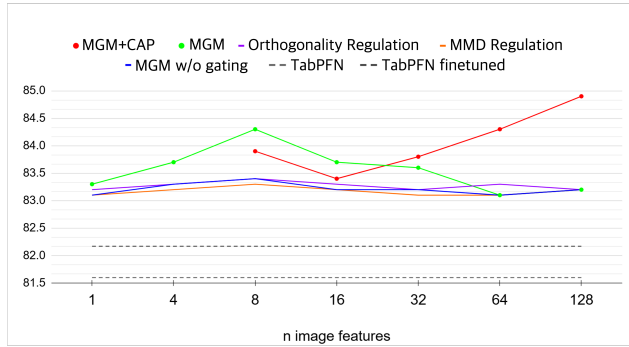


Figure S2. **Additional results of token-count and attention mass.** We extend the analysis in ?? to additional datasets. For Mass and Calc, each sample contains three images, and thus the number of non-tabular tokens is given by the number of mgm heads times 3. The number of tabular tokens is 25, 4 and 4 for Airbnb, Mass, and Calc, respectively.



(a) Image-only MGM variants.



(b) Full MMPFN (tabular + image) variants.

Figure S3. **Effect of distribution-alignment and orthogonality regularization.** Under the same experimental conditions as ??, we compare the performance of variants that incorporate orthogonality and MMD-based regularization constraints to the MGM baseline, for both image-only and tabular+image settings.

the MGM baseline, and incorporating the same loss directly into MGM also fails to yield improvements. Together with our cosine-similarity analysis, these negative results suggest

that embeddings extracted by MGM from image and text modalities are already mapped into a semantically compatible space with tabular embeddings, enabling effective interaction through the attention module. Moreover, while MMD-style losses can reduce distributional gaps, they may also suppress discriminative variations, leading to performance degradation. We therefore infer that once MGM and CAP produce sufficiently aligned embeddings, enforcing additional distributional alignment does not provide further gains and can even be detrimental.

Comparison with Patch-Token Features. In MMPFN, the text encoder [?] and the image encoder [?] produce output embeddings for every token (e.g., text tokens or image patches). In principle, one could replace the [CLS]-based MGM features with ViT patch-token outputs and use all token embeddings directly for feature generation. However, this design has both practical and empirical drawbacks. From a memory perspective, using all patch tokens is substantially more expensive than using the aggregated [CLS] token. For example, when resizing PAD-UFES-20 images to $336 = 14 \times 24$ pixels and encoding them with the DINOv2 ViT-B/14 backbone, the model produces 576 token embeddings per image—more than four times the number of MGM heads (128) used in our experiments. In terms of storage, the [CLS] embedding requires only 7.1 MB, whereas retaining all patch-token outputs occupies 4.1 GB, leading to a prohibitive increase in memory consumption. A similar issue arises for text: in the Cloth dataset, many text attributes approach the maximum input length of 512 tokens, so storing all token embeddings again results in excessive memory usage.

Empirically, we also observe that models using ViT

patch-token outputs underperform those relying on the $[CLS]$ -driven MGM features. On PU20, replacing the $[CLS]$ -based MGM heads with patch-token features decreases accuracy by 0.85 percentage points (from 85.22% to 84.02%). We attribute this degradation to the fact that the $[CLS]$ representation of a well-trained foundation model already encodes task-relevant global information, while raw token-level outputs contain substantial redundancy and noise. Consequently, patch-token features are less suitable than $[CLS]$ -based MGM heads for constructing compact, tabular-like feature representations.

Additional qualitative results of attention imbalance.

To complement the analysis in the main paper, we further examine the relationship between token count and attention mass on additional datasets. As shown in Figure S2, we observe a consistent trend across all datasets: as the number of non-tabular tokens increases, the attention mass assigned to the non-tabular modality increases monotonically, while the attention to tabular tokens decreases accordingly.

For datasets such as Mass and Calc, each sample contains multiple images, and the number of non-tabular tokens is given by the number of MGM heads multiplied by the number of images. Despite variations in the number of tabular tokens across datasets, such as 25 for Airbnb and 4 for Mass and Calc, the same qualitative behavior consistently emerges. These results further indicate that attention allocation is primarily determined by the relative proportion of tokens within the input sequence.

Implementation Details. For the training procedure, we adopt the official TabPFN repository¹ and modify it to support MMPFN by extending the `MMPFNClassifier` and `MMPFNRegressor` classes. Fine-tuning is performed by splitting the available data into training and validation sets and updating model parameters based on the validation loss. We use a small learning rate of 1×10^{-5} , a fixed budget of 100 training steps, and `ScheduleFree` [?] for learning rate scheduling. Figure S4 shows the learning curve on four different datasets. For contrastive-pretraining baselines [? ? ?], we train for 500 epochs using a cosine-annealing scheduler with a 10-epoch warmup.

Text Data Pre-Processing. We adopt the text pre-processing pipeline of TTT [?], following the implementations provided in the official codebase. For the Salary dataset, the original source URL referenced in [?] is no longer accessible, so we use a Kaggle-hosted copy instead. Applying the official TTT scripts to this version does not

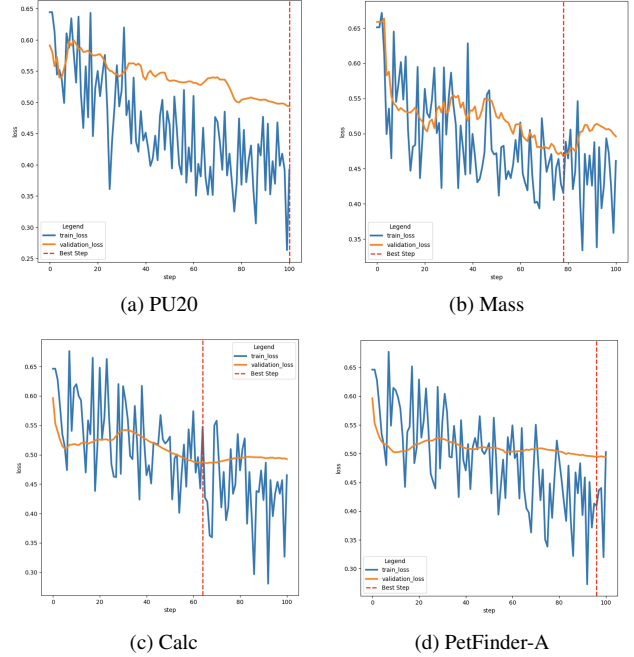


Figure S4. **Fine-tuning loss curves of MMPFN across datasets.** Each subfigure shows the validation loss over training steps for four different datasets, illustrating stable convergence behavior under our fine-tuning setup.

reproduce the exact dataset size reported in the paper, suggesting minor discrepancies. We therefore re-evaluate the TTT baseline on this revised dataset; the resulting accuracy (46.5) closely matches the originally reported performance, confirming that the new version is suitable for evaluating our model.

Both ELECTRA and DeBERTa text encoders are limited to 512 input tokens, so longer sequences are truncated. For datasets with multiple text attributes, we extract embeddings for each attribute separately and incorporate them as additional text features, whereas TTT concatenates all text columns into a single sequence; this yields small but consistent accuracy gains, although the improvements remain within the error margin and are not reported in the main comparison table. The Airbnb and PetFinder datasets contain Chinese characters in their text fields; because the ELECTRA variant we use is not pretrained on Chinese, these characters are replaced with empty strings before encoding. For CatBoost and AutoGluon, we rely on the libraries’ built-in text handling capabilities.

Cosine Similarity Computation. We provide the detailed procedure used for the cosine-similarity-based correlation analysis in [?]. The TabPFN encoder for tabular data normally groups multiple features into a single embedding to reduce memory usage, but such grouping can intro-

¹https://github.com/LennartPurucker/finetune_tabPFN_v2

duce noise when comparing cosine similarity between tabular and image (or text) embeddings. To obtain more precise relationships, we set the tabular group size to 1, generate an individual embedding for each feature, and then compare these feature-wise embeddings with image embeddings.

Cosine similarity between input features is computed at the instance level and subsequently averaged across instances. Averaging embeddings over the entire dataset before computing similarity vectors would be cheaper but tends to underrepresent the contribution of individual samples, whereas computing similarity for every token embedding is prohibitively expensive and makes global patterns difficult to visualize. We therefore adopt an intermediate strategy, which balances computational cost and fidelity.