A `MultiModN` Framework Implementation

A **task- and modality-agnostic open-source framework** `MultiModN` solution has been implemented in Python using PyTorch as the primary machine learning framework. The `/multimodn` package contains the `MultiModN` model and its components. The `/datasets` package is responsible of preparing the data inputs for `MultiModN`. Some examples using the public Titanic dataset have been provided.

The code is available at: `https://github.com/epfl-iglobalhealth/MultiModN`

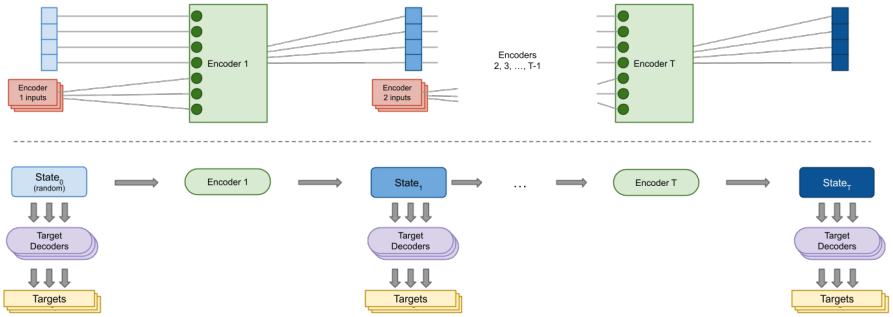

Figure 6: **Architecture of `MultiModN` code**. The **upper panel** shows a more detailed depiction of sequential encoding using a series of model-agnostic encoders which receive inputs of variable dimension to create the evolving state vector, which represents the shared feature space. The **lower panel** shows how each state can be probed by any number of target decoders.

A.1 `MultiModN` metrics

During training and evaluation, the metrics of the model are stored in a log at each epoch in a matrix of dimensions $(E+1) * D$, where E is the number of encoders and D the number of decoders. Each row represents the metrics for a target at each state of the model.

A.2 Code structure

A.2.1 MultiModN

`/multimodn` package contains the `MultiModN` model and its modules:

[1] Encoders: `/multimodn/encoders`
[2] Decoders: `/multimodn/decoders`
[3] State: `/multimodn/state.py`

A.2.2 Datasets

`/dataset` package contains the MultiModDataset abstract class, compatible with `MultiModN`.

Specific **datasets** are added in the `/dataset` directory and must fulfill the following requirements:

- Contain a dataset class that inherit MultiModDataset or has a method to convert into a MultiModDataset
- Contain a `.sh` script responsible of getting the data and store it in `/data` folder

`__getitem__` function of MultiModDataset subclasses must yield elements of the following shape:

```
tuple
(
    data: [torch.Tensor],
    targets: numpy.ndarray,
```

```
    (optional) encoding_sequence: numpy.ndarray
)
```

namely a tuple containing an array of tensors representing the features for each subsequent encoder, a numpy array representing the different targets and optionally a numpy array giving the order in which to apply the encoders to the subsequent data tensors. Note: `data` and `encoding_sequence` must have the same length.

Missing values. The user is able to choose to keep missing values (`nan` values). Missing values can be present in the tensors yielded by the dataset and are managed by `MultiModN`.

A.2.3 Pipelines

`/pipeline` package contains the training pipelines using `MultiModN` for Multimodal Learning. It follows the following steps:

- Create `MultiModDataset` and the `dataloader` associated
- Create the list of encoders according to the features shape of the MultiModDataset
- Create the list of decoders according to the targets of the MultiModDataset
- Init, train and test the `MultiModN` model
- Store the trained model, training history and save learning curves

A.3 Quick start

Quick start running `MultiModN` on Titanic example pipeline with a Multilayer Perceptron encoder:

```
./datasets/titanic/get_data.sh
python3 pipelines/titanic/titanic_mlp_pipeline.py
```

Open `pipelines/titanic/titanic_mlp.png` to look at the training curves.

B Additional details about `MultiModN` Architecture

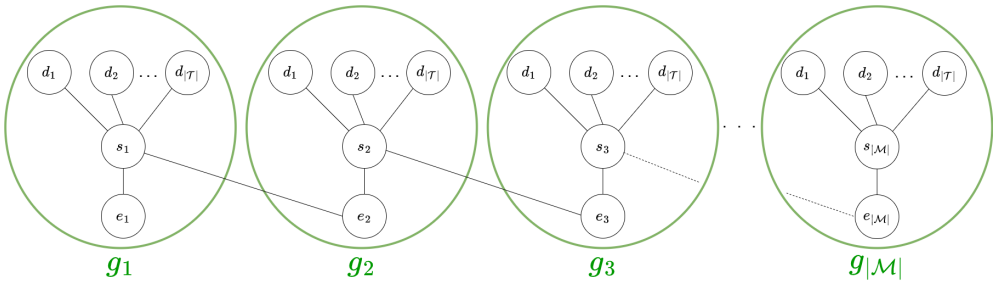

Figure 7: **Schematic representation of the *modules* (g, groups) of** `MultiModN`. e: encoders, s: state vector, d: decoders. Each module is connected by a single edge between s_n and e_{n+1}. There are $|\mathcal{M}|$ groups (i.e. input-specific modules) and $|\mathcal{T}|$ decoders per module.

Modularity. In the following, we detail the computation of `MultiModN`'s modularity measure. The total number of edges in a `MultiModN` module is $|\mathcal{T}| + 1$. The total number of modules is $|\mathcal{M}|$ and there are $|\mathcal{M}| - 1$ edges connecting consecutive modules, which makes for a total number of edges in the entire `MultiModN` model of $m = |\mathcal{M}|(|\mathcal{T}| + 2) - 1$.

To compute the modularity following using the formalization proposed by Newman et al [29], we need to define groups. In the case of `MultiModN`, each group corresponds to one *module*. Let G be the matrix whose components g_{ij} is the fraction of edges in the original network that connect vertices in group i to those in group j.

Within `MultiModN`, each group contains $(|\mathcal{T}|+1)$ edges and is connected to adjacent groups by two edges, with the exception of g_1 and $g_{|\mathcal{M}|}$, which are connected to only one other group (cf. Figure 7).

Thus, G is a tridiagonal matrix whose diagonal elements G_{ii} are equal to $(|\mathcal{T}|+1)/m$ and whose upper and lower diagonal elements are equal to $1/m$:

$$G = \frac{1}{m}\begin{pmatrix} |\mathcal{T}|+1 & 1 & & & \\ 1 & |\mathcal{T}|+1 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & |\mathcal{T}|+1 & 1 \\ & & & 1 & |\mathcal{T}|+1 \end{pmatrix}$$

The modularity measure is defined by $\mathcal{Q} = \text{tr}(G) - \|G^2\|$, where $\text{tr}(G)$, is the trace of G and $\|G^2\|$ is the sum of elements of G^2. The trace of G is equal to $\frac{|\mathcal{M}|}{m}(\mathcal{T}+1)$ and we have G^2 equal to:

$$\frac{1}{m^2}\begin{pmatrix} (|\mathcal{T}|+1)^2+1 & 2(|\mathcal{T}|+1) & 1 & & & \\ 2(|\mathcal{T}|+1) & (|\mathcal{T}|+1)^2+2 & 2(|\mathcal{T}|+1) & 1 & & \\ 1 & 2(|\mathcal{T}|+1) & (|\mathcal{T}|+1)^2+2 & 2(|\mathcal{T}|+1) & 1 & \\ & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & 1 & 2(|\mathcal{T}|+1) & (|\mathcal{T}|+1)^2+2 & 2(|\mathcal{T}|+1) \\ & & & 1 & 2(|\mathcal{T}|+1) & (|\mathcal{T}|+1)^2+1 \end{pmatrix}$$

Hence, we have:

$$\begin{aligned} \|G^2\| &= \frac{1}{m^2}\left[|\mathcal{M}|((|\mathcal{T}|+1)^2+2) - 2 + 2(|\mathcal{M}|-1)2(|\mathcal{T}|+1) + 2(|\mathcal{M}|-2)\right] \\ &= \frac{|\mathcal{M}||\mathcal{T}|^2 + 6|\mathcal{M}||\mathcal{T}| + 9|\mathcal{M}| - 4|\mathcal{T}| - 10}{|\mathcal{M}|^2|\mathcal{T}|^2 + 4|\mathcal{M}|^2|\mathcal{T}| + 4|\mathcal{M}|^2| - 2|\mathcal{M}||\mathcal{T}| - 4|\mathcal{M}| + 1} \end{aligned}$$

It is important to note that when $\mathcal{M}$ increases, the trace will tend to $(|\mathcal{T}|+1)/(|\mathcal{T}|+2) \approx 1$ for a large number of tasks. Moreover, when $|\mathcal{M}|$increases $||G^2||$ will decrease towards to 0. Thus, for a large number of tasks, we have a modularity measure that increases and tends to 1 with the number of modalities.

Architecture alignment between `P-Fusion` and `MultiModN`. We purposely align the feature extraction pipelines of `P-Fusion` and `MultiModN` in order to best isolate the effect of monolithic-parallel fusion vs. modular-sequential fusion. In Figure 8, we see how the alignment is limited to the input, where both `MultiModN` and `P-Fusion` share the feature-extraction of each modality, where the `MultiModN` encoders receive an embedding (emb). No element of `MultiModN` is changed as described in Figure 1. Embeddings from missing data can be skipped (i.e. not encoded) by `MultiModN`.

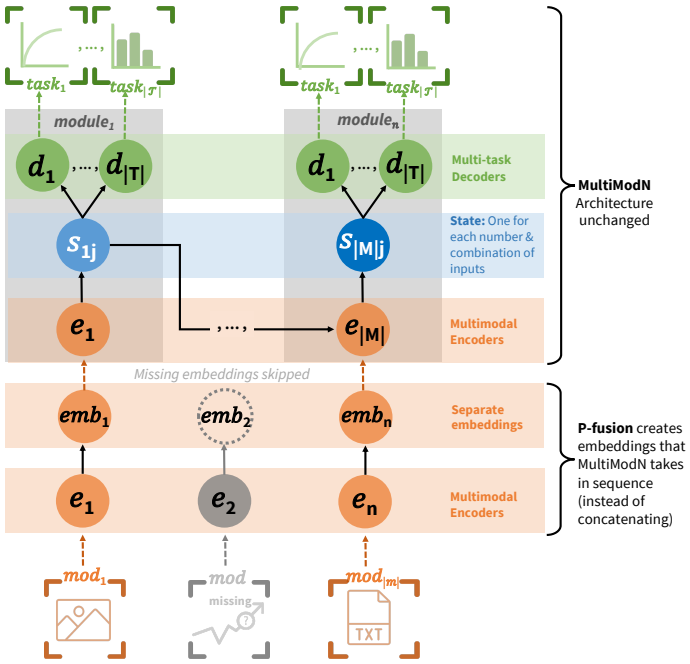

Figure 8: **Alignment of `P-Fusion` and `MultiModN` architectures.** We purposely ensure that the feature extraction pipelines are aligned between `P-Fusion` and `MultiModN`. To this end, we use the embeddings (emb) produced by `P-Fusion` as inputs into the `MultiModN` encoders (e). No element of `MultiModN` is changed. `MultiModN` encoders (e) are in orange, the `MultiModN` state (s) is in blue and multi-task `MultiModN` decoders (d) are in green.

For MIMIC, feature extraction for each modularity is replicated from previous work [20] and we use embeddings generated from a set of pre-trained models. For Weather and Education, as there were no pre-existing embedding models, we design autoencoders trained to reconstruct the original input features from a latent space. We keep the autoencoder's encoder and decoder structure exactly aligned with the encoders of `MultiModN` (two ReLU activated, fully-connected Dense layers and a third layer either generating the state representation with a ReLU activation or the final prediction with a sigmoid activation). We also align the number of trainable parameters with `MultiModN`'s modality-based encoders for a fair baseline comparison by selecting an appropriate state representation size per modality to equal the state representation in `MultiModN`. The remaining hyperparameters are left exactly the same (batch size, hidden layer size, dropout rate, optimization metrics, loss function).

C Datasets and Tasks

A description of all tasks is provided in Table 2. In the following paragraphs, we detail the preprocessing decisions on the datasets for context and reproducibility.

MIMIC. The data includes four input modalities (tabular, textual, images, and time-series) derived from several sources for each patient. We align our preprocessing pipeline exactly with the study from which our baseline of `P-Fusion` is derived [20] (described in 4.1). To this end, we use patient-level feature embeddings extracted by the pre-trained models described in [20] and depicted in Figure 8.

The dataset is a combination of two MIMIC databases: MIMIC-IV v.1.0 [21] and MIMIC-CXR-JPG v.2.0.0 [22]. After gaining authorized access to PhysioNet online repository [30], the embedding dataset can be downloaded via this link: `https://physionet.org/content/haim-multimodal/1.0.1/`. The dataset comprises $45,050$ samples, each corresponding to a time point during a patient's hospital stay when a chest X-ray was obtained. It covers a total of $8,655$ unique patient stays. To ensure data quality and limit our experiments to two thematic tasks (diagnosis of $task_1$: cardiomegaly and $task_2$: enlarged cardiomediastinum), we remove duplicates (based on image id and image acquisition time), and retain only relevant samples that have valid labels for both targets of interest,

i.e. both $task_1$ and $task_2$ are either present (1) or absent (0). Subsequent experiments for these tasks are thus performed on the $921/45,050$ selected relevant patients.

EDU. This dataset involves hand-crafted features extracted for $5,611$ students across 10 weeks of data. The preprocessing of data is an exact replication of several related works using the same dataset [34, 31, 42] based on 4 feature sets determined as predictive for MOOC courses in [43]. 45 features regarding problem and video data are extracted per student per week, covering features like *Delay Lecture*, which calculates the average delay in viewing video lectures after they are released to students or *TotalClicksProblem*, the number of clicks that a student has made on problems this week. The features are normalized with min-max normalization, and missing values are imputed with zeros that have meaning i.e. no problem events in a week is correctly inferred as zero problems attempted. In this setting, missingness is a valued, predictive feature of the outcome, and thus we do not perform missingness experiments on this dataset. `MultiModN` has the ability to select whether missingenss is encoded or not, and thus it would not suffer a disadvantage in a setting where missingness should be featurized.

In MOOCs, a common issue is that students join a course and never participate in any assignments, homeworks, or quizzes. This could be due to registering aspirationally, to read some material, or to watch videos [44]. An instructor can easily classify students who have never completed an assignment as failing students. As introduced by Swamy et al. in [34] and used with this dataset in related work [31, 42, 33], EDU has removed students that were predicted to fail in the first two weeks simply by having turned in no assignments (99% confidence of failing with an out-of-the-box logistic regression model, where the confidence threshold was tuned over balanced accuracy calculations). It has been shown that including these students will artificially increase the performance of the model, providing even better results than those showcased by `MultiModN` in this work [34]. We thus exclude these students to test a more challenging modeling problem.

Weather. The Weather2k dataset, presented in [36], covers features from $1,866$ weather stations with 23 features covering seven different units of measurements (degrees, meters, HPA, celsius, percentage, ms^{-1}, millimeters). To align these features on vastly different scales, we normalize the data. We use the large extract (R) provided by the authors instead of the smaller representative sample also highlighted in the benchmark paper (S) [36]. We use the first 24 hourly measurements as input to train the `MultiModN` model and calculate the five regression tasks as determined in Table 2 below.

Tasks. As showcased in Table 2, our evaluation covers 10 binary and regression tasks in two settings: static (one value per datapoint) or continuous (changing values per datapoint, per timestep).

	Task	Type	Name	Description
MIMIC	**1**	Static Binary	Cardiomegaly	Labels determined as per [20] using NegBio [45] and CheXpert [46] to process radiology notes, resulting in four diagnostic outcomes: positive, negative, uncertain, or missing.
	2	Static Binary	Enlarged Cardiomediastatinum	Labels determined as per [20] using NegBio [45] and CheXpert [46]. The set of label values is identical to the one for cardiomegaly.
EDU	**3**	Static Binary	Student Success Prediction	End of course pass-fail prediction (per student) as per [34] on the course.
	4	Continuous Binary	Student Dropout Prediction	1 if student has any non-zero value on a video or problem feature from next week until the end of the course, 0 if not. Not valid for the last week, so the task involves n-1 decoder steps for n timesteps. Can be easily extended to a multiclass task by separating video or problem involvement until the end of the course into separate classes.
	5	Continuous Regression	Next Week Performance Forecasting	Moving average (per student, per week) of three student performance features from [43] and removed in baseline paper [34]: *Student Shape* (recieving the maximum quiz grade on the first attempt), *CompetencyAlignment* (number of problems the student has passed this week), *CompetencyStrength* (extent to which a student passes a quiz getting the maximum grade with few attempts).
Weather	**6**	Continuous Regression	Short Term Temperature Forecasting	Changing air temperature measurements (collected per station, per hour), shifted by 24 hourly measurements (1 day).
	7	Continuous Regression	Mid Term Temperature Forecasting	Changing air temperature measurements (collected per station, per hour), shifted by 72 hourly measurements (3 days).
	8	Continuous Regression	Long Term Temperature Forecasting	Changing air temperature measurements (collected per station, per hour), shifted by 720 hourly measurements (30 days).
	9	Static Regression	Relative Humidity	Instantaneous humidity relative to saturation as a percentage at 48h from 2.5 meters above the ground, as used as a benchmark forecasting task in [36].
	10	Static Regression	Visibility	10 minute mean horizontal visibility in meters at 48 hr from 2.8 meters above the ground, as used as a benchmark forecasting task in [36].

Table 2: Description of all 10 tasks used to evaluate `MultiModN`.

Note that for $task_4$, the three features used to calculate next week's performance were not included in the original input features because of possible data leakage, as student performance on quizzes directly contributes to their overall grade (Pass/Fail).

For the AUROC curves on $tasks_{9-10}$ in Figure 2, we conduct a binarization of the two last regression tasks. To align with the regression task, we conduct a static forecasting prediction per station for the relative humidity or visibility with a time window of 24h for the 48th timestep (one day in advance). While `MultiModN` is capable of making a prediction at each continuous timestep, `P-Fusion` is not able to do this without a separate decoder at each timestep, and therefore to compare the tasks we must choose a static analysis. We choose a threshold based on the normalized targets: 0.75 for humidity and 0.25 for visibility (selected based on the distribution of the feature values for the first 1000 timesteps), and evaluate the predictions as a binary task over this threshold.

Analogously, in the Section E analysis below on the binarization of the remaining regression tasks, we express the continuous regression tasks for temperature forecasting $tasks_{6-8}$ as a static binary task. To do this, we evaluate the prediction from the full window (24th timestep) at the respective forecasting timestep (48, 96, 744). The threshold we select is 0.3, closely corresponding to the normalized mean of the temperature (0.301) over the first 1000 timesteps.

D Model Optimization

Table 3 indicates the chosen hyperparameters for the experiments conducted in Section 6 of the paper, selected based on the optimal hyperparameters for the multi-task settings (all the tasks for a dataset predicted jointly). The single tasks use the same hyperparameters as the multi-task settings. For saving the best model across training epochs in the time series settings (EDU, Weather), our optimization metric saved the ones with best validation set results on the most short term task. Therefore, for EDU, we use MSE of $task_5$, predicting next week performance, and $task_6$ for Weather, forecasting temperature within a day). The intuition is that choosing the best model on the short term task would allow the model to emphasize stronger short-term connections, which in turn would improve long term performance.

Task	# of Timesteps	Batch Size	Dropout Rate	Hidden Layer Size	State Rep. Size	Save Best Model (chosen metric)
1	1	16	0.2	32	50	$task_1$ Val BAC + Macro AUROC
2	1	16	0.2	32	50	$task_2$ Val BAC + Macro AUROC
MIMIC	1	**16**	**0.2**	**32**	**50**	$tasks_{1-2}$ **BAC + Macro AUROC**
3	10	64	0.1	32	20	$task_3$ Val Accuracy
4	10	64	0.1	32	20	$task_4$ Val Accuracy
5	10	64	0.1	32	20	$task_5$ Val MSE
EDU	**10**	**64**	**0.1**	**32**	**20**	$task_5$ **Val MSE**
6	24	128	0.1	32	20	$task_6$ Val MSE
7	24	128	0.1	32	20	$task_7$ Val MSE
8	24	128	0.1	32	20	$task_8$ Val MSE
9	24	128	0.1	32	20	$task_9$ Val MSE
10	24	128	0.1	32	20	$task_{10}$ Val MSE
Weather	**24**	**128**	**0.1**	**32**	**20**	$task_6$ **Val MSE**

Table 3: Hyperparameters selected for each experiment. We tuned the hyperparameters for the multi-task models (MIMIC, EDU, Weather) and used the same hyperparameters for each single-task model for a fair comparison.

In Figure B, we examine a case study of `MultiModN`'s changing performance on $task_3$, student success prediction in the EDU dataset, by varying hyperparameters across three model architectures (chosen for small, medium, and large hyperparameter initializations). We note that batch size is fairly robust across all three initial model settings, with large batch size on the largest model having slightly variable performance. Examining changing dropout rate, we note that with medium and large models, change in dropout impacts performance considerably. This allows us to hypothesize that high dropout on larger state representations does not allow the model to learn everything it can from the data.

Looking at varied hidden layer size, we see comparable performance for the small and medium initializations, but note that in the large case, having a smaller hidden layer size is important to maintain performance. Even as `MultiModN` performance trends upwards with larger hidden layer size (i.e. 128) for the large initialization, the confidence interval is large, so performance is not stable. Lastly, observing state representation size, we see that when state representation is too small for the task (i.e. 1, 5), the small and large models are adversly impacted. Additionally, when state representation is too large (i.e. 100), performance seems to drop or increase variability again. It is therefore important to tune `MultiModN` and find the right state representation size for the dataset and predictive task(s).

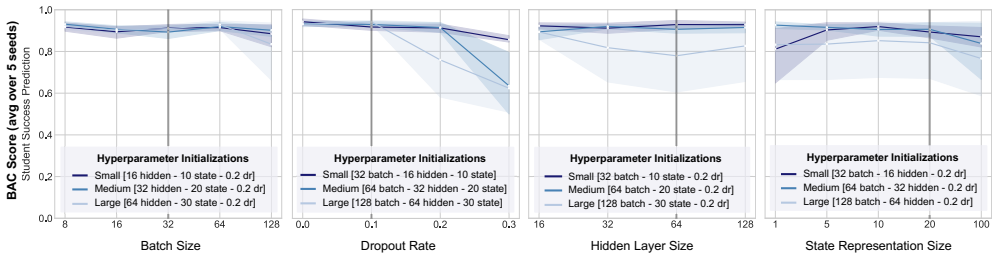

Figure 9: `MultiModN` hyperparameter selection across four parameters on $task_3$ of the EDU dataset (Pass/Fail). Each individual parameter is varied on the x-axis (***dr**: dropout rate*) with all other initializations fixed (grouped in small, medium, and large values). These are compared in terms of balanced accuracy (BAC). 95% CIs are shaded.

Experimental Setup. For the results reported in Sections 6.1, 6.2 and 6.4, we perform 5-fold stratified cross-validation with 80-10-10 train-validation-test split. Due to the time-series nature of the EDU and Weather datasets, we orient the stratification the real labels associated with the longest-term task. For EDU, $task_3$ (end of course pass-fail prediction) and for Weather $task_8$ (30-day temperature forecasting) is chosen for the stratification split.

In an alternative approach, regarding the MIMIC dataset, a two-step procedure was implemented to address the imbalanced class ratios, given the absence of a prioritized task. Initially, a new dummy label was assigned to each sample, indicating positivity if both pathologies are present and negativity otherwise. Subsequently, a label was assigned to each unique hospital stay based on the aggregated labels from the first step. A hospital stay was considered positive if the number of times sample from that stay has been found positive is greater than or equal to the half of samples with the same hospital stay ID. The latter, as outlined in [20], ensured that no information was leaked on the hospital stay level during stratification.

Experiments were conducted using the same architecture in PyTorch (MIMIC) and TensorFlow (EDU, Weather), to provide multiple implementations across training frameworks for ease of use. We use an Adam optimizer with gradient clipping across all experiments.

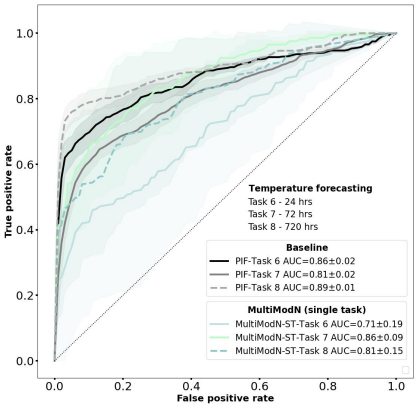

Figure 10: AUROC for three additional binary prediction tasks in Weather2k. Targets predicted by `P-Fusion` are compared to `MultiModN`. 95%CIs are shaded.

E Additional Experiments

E.1 Single task

We present the binarized results (AUROC curves) for several additional regression tasks ($tasks_{6-8}$ for Weather) in Figure 10. The specific details of binarization are discussed above in Appendix

Section C. We note that the confidence intervals overlap for `P-Fusion` and `MultiModN` over all Weather tasks. However, it is clear that the performance of `MultiModN` varies a lot (large CIs). This could be due to the design of the binarization as originally in the benchmark paper [36], this was introduced as a regression task. Another contributing factor to large CIs could be that the model was trained across all timesteps but only evaluated on one timestep for a comparable binarization. Despite these caveats, we can statistically conclude that `P-Fusion` and `MultiModN` performance are comparable on these additional tasks.

E.2 Interpretability

We perform an interpretability analysis for the EDU dataset, analogous to the local and global interpretability analysis on MIMIC from Figure 4. The global analysis (IMC) is conducted over all students for the first week of the course. We note interesting findings: specifically that problem interactions are more important for $tasks_{3-4}$ while video interactions are more important for $task_5$.

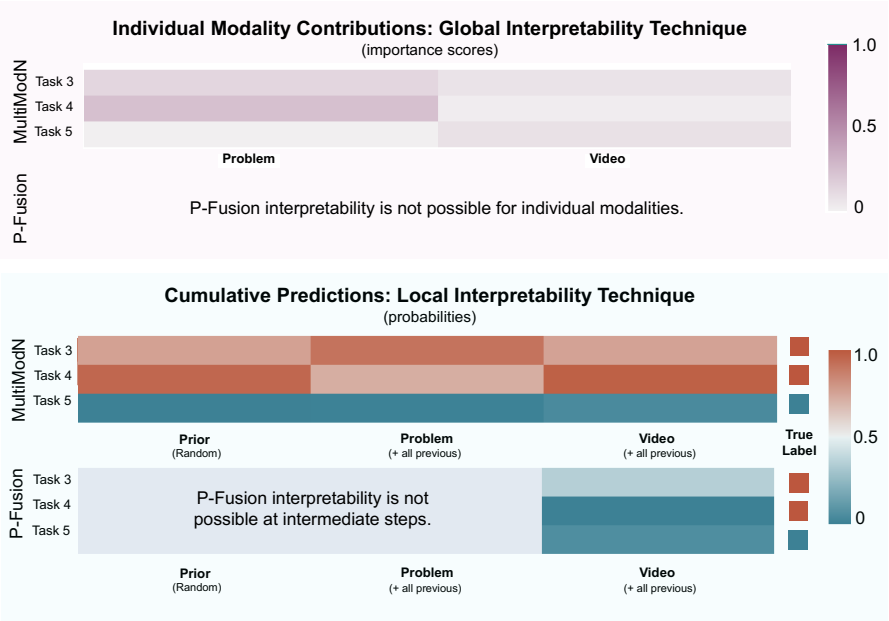

Figure 11: **Inherent modality-specific model explainability in `MultiModN` for $tasks_{3-5}$.** Heatmaps show individual modality contributions (IMC) **(top)** and cumulative contributions (CP) **(bottom)**: respectively **importance score** (global explainability) or **cumulative probability** (local explainability). The multi-task `MultiModN` for $task_{3-5}$ in EDU is compared to two single-task `P-Fusion` models. IMC are only possible for `MultiModN` (only 1 modality encoded, rest are skipped). CP are made sequentially from states encoding all previous modalities. `P-Fusion` is unable to naturally decompose modality-specific contributions (can only make predictions once all modalities are encoded). IMC is computed across all students in the test set. CP is computed for a single student, (true label = 1 for $tasks_{3-4}$ and 0 for $task_5$). The CP heatmap shows probability ranging from **confident negative diagnosis (0)** to **perfect uncertainty** and **confident positive diagnosis (1)** .

The student selected for the CP local analysis passes the course (1 for $task_3$) and does not dropout (1 for $task_4$), but does not have strong performance in the next week (~ 0 in $task_5$). This analysis is also conducted across the first week of course interactions. We see that `P-Fusion` cannot produce modality-specific interpretations and predicts the incorrect label. However, `MultiModN` is able to identify a changing confidence level across modalities, eventually ending on the right prediction for all tasks. The confidence for $task_3$ increases with the student's problem interactions and reduces for their video interactions. This could have potential for designing an intervention to improve student learning outcomes. We note that for $task_5$, both the students' problem and video interactions contribute similarly to the prediction.

E.3 Missingness

We expand on the missingness experiments presented in 6.4. Here, we present further control experiments (training on data missing-at-random, MAR) in both MIMIC tasks ($task_1$: diagnosis of Cardiomegaly 12 and $task_2$: diagnosis of Enlarged Cardiomediastinum 13).

In the first two subplots of each figure, both `P-Fusion` (black) and `MultiModN` (red) are trained on MNAR data and then evaluated either on a test set at risk of catastrophic failure (where the pattern of MNAR is label-flipped, first figure) or on a test set with no missingness. As can be seen in the first figures, `P-Fusion` suffers catastrophic failure in the MNAR flip, becoming worse than random when a single modality is missing at 80%, as opposed to `MultiModN`, which only decreases AUROC about 10%. When the test set has no missing values, `P-Fusion` and `MultiModN` are not significantly different, proving that the catastrophic failure of `P-Fusion` is due to MNAR. This is further confirmed in the last two plots of each task, where the models are trained on MAR data and evaluated on test sets either without missing values or MAR missingness.

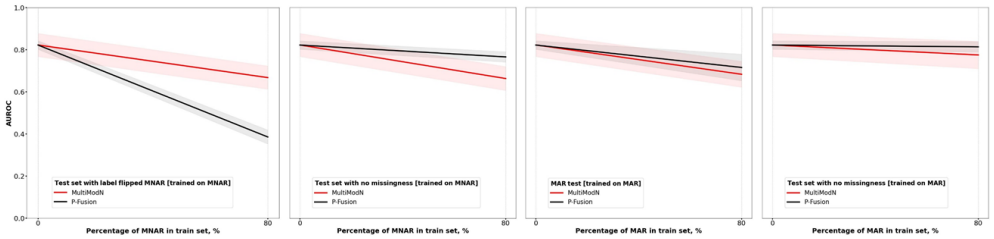

Figure 12: **Detailed missingness experiments for $task_1$ (Cardiomegaly).** `P-Fusion` (black) and `MultiModN` (red) are trained on MIMIC data where various percentages of a single modality are missing (0 or 80%) either for a single class (MNAR, first two plots) or without correlation to either class (MAR, last two plots). The AUROCs are shown for each when evaluated on test sets which either have a risk of catastrophic failure (first plot, MNAR with label flip) or on test sets without missingness or MAR missingness. CI95% shaded.

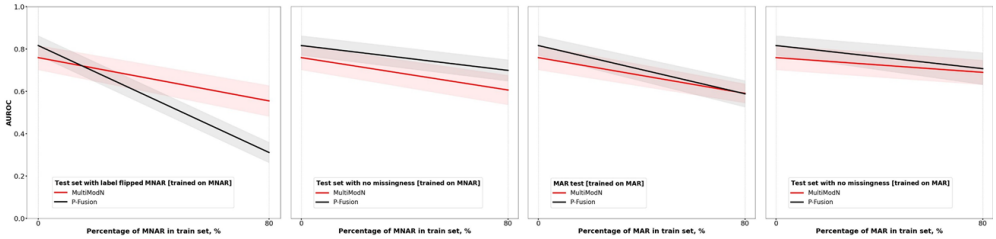

Figure 13: **Detailed missingness experiments for $task_2$ (Enlarged Cardiomediastinum).** `P-Fusion` (black) and `MultiModN` (red) are trained on MIMIC data where various percentages of a single modality are missing (0 or 80%) either for a single class (MNAR, first two plots) or without correlation to either class (MAR, last two plots). The AUROCs are shown for each when evaluated on test sets which either have a risk of catastrophic failure (first plot, MNAR with label flip) or on test sets without missingness or MAR missingness. CI95% shaded.

E.4 Comparison to a `P-Fusion` Transformer

Additional experiments with a Transformer have been conducted on 10 tasks across three datasets. Results are showcased below in two tables (left for $tasks_{1-4}$ and right for $tasks_{5-10}$) with 95% CIs.

The hyperparameter-tuned architecture (based on head size, number of transformer blocks, MLP units) for EDU and Weather is a transformer model with 4 transformer blocks, 4 heads of size 256, dropout of 0.25, MLP units 128 with dropout 0.4, batch size 64, trained for 50 epochs with cross-entropy loss. For MIMIC, the most performant (tuned) transformer architecture includes 2 transformer blocks with 3 heads of size 128, MLP units 32 with batch size 32. We train this architecture on each decoder

task individually and all tasks together for a total of 13 new models with the exact preprocessing steps as in the `P-Fusion` and `MultiModN` experiments. The results indicate that `MultiModN` often outperforms or at least matches the `P-Fusion` Transformer benchmark in the vast majority of single task and multi-task settings, and comes with several interpretability, missingness, and modularity advantages. Specifically, using the primary metric for each task (BAC for the classification tasks and MSE for the regression tasks), `MultiModN` beats the Transformer baseline significantly in 7 tasks, overlaps 95% CIs in 11 tasks, and loses very slightly (by 0.01) in 2 regression tasks.

Classification

	Task	Encoder	BAC	F1
MIMIC	Cardiomegaly	Single	0.50 ± 0.01	0.43 ± 0.02
		Multi	0.50 ± 0.02	0.43 ± 0.03
	ECM	Single	0.50 ± 0.02	0.44 ± 0.03
		Multi	0.50 ± 0.01	0.42 ± 0.01
EDU	Success (P/F)	Single	0.50 ± 0.01	0.40 ± 0.01
		Multi	0.96 ± 0.01	0.95 ± 0.02
	Dropout	Single	0.83 ± 0.22	0.83 ± 0.24
		Multi	0.28 ± 0.02	0.27 ± 0.02

Regression

	Task	Encoder	MSE	R2
EDU	Weekly Perf.	Single	0.01 ± 0.01	0.21 ± 0.06
		Multi	3.91 ± 3.49	-649.05 ± 506.01
WEATHER	Temp. (24h)	Single	0.00 ± 0.01	0.80 ± 0.10
		Multi	0.00 ± 0.01	0.75 ± 0.08
	Temp. (72h)	Single	0.00 ± 0.01	0.71 ± 0.18
		Multi	0.00 ± 0.01	0.77 ± 0.08
	Temp. (720h)	Single	0.00 ± 0.01	0.82 ± 0.13
		Multi	0.00 ± 0.01	0.78 ± 0.03
	Humidity	Single	0.01 ± 0.01	0.69 ± 0.22
		Multi	0.03 ± 0.01	0.00 ± 0.41
	Visibility	Single	0.02 ± 0.01	0.82 ± 0.11
		Multi	0.05 ± 0.02	0.50 ± 0.12

Figure 14: **Performance of the `P-Fusion` Transformer on 10 classification and regression tasks across 3 datasets.** Results are showcased with 95% confidence intervals. BAC and MSE are the primary evaluation metrics for classification and regression respectively.

E.5 Additional Inference Settings

Task	Modalities (Inference)	MultiModN	P-Fusion
CM	|demo|	0.50	0.50
	|text|	0.61	0.58
	|ts|	0.52	0.53
	|visual|	0.75	0.75
ECM	|demo|	0.50	0.51
	|text|	0.59	0.52
	|ts|	0.53	0.54
	|visual|	0.71	0.72
CM	|demo|text|	0.55	0.57
	|demo|ts|	0.54	0.52
	|demo|visual|	0.74	0.76
	|text|ts|	0.61	0.59
	|visual|text|	0.74	0.74
	|visual|ts|	0.76	0.76
ECM	|demo|text|	0.56	0.52
	|demo|ts|	0.47	0.56
	|demo|visual|	0.69	0.71
	|text|ts|	0.60	0.51
	|visual|text|	0.71	0.68
	|visual|ts|	0.72	0.71

Task	Modalities (Inference)	MultiModN	P-Fusion
CM	|demo|text|ts|	0.64	0.60
	|demo|visual|text|	0.75	0.75
	|demo|visual|ts|	0.77	0.76
	|visual|text|ts|	0.74	0.75
ECM	|demo|text|ts|	0.56	0.51
	|demo|visual|text|	0.72	0.68
	|demo|visual|ts|	0.73	0.70
	|visual|text|ts|	0.71	0.67
ECM CM	|demo|visual|text|ts|	0.74	0.75
	|demo|visual|text|ts|	0.71	0.68

Figure 15: **Detailed modality inference experiments for `MultiModN` in comparison to `P-Fusion`.** In these experiments, different combinations of modalities and orderings at the time of inference are used for the two tasks in the MIMIC dataset. All 95% CIs overlap between the two models.

To provide insight into performance gains, we performed additional experiments to showcase the benefits of modularity with vastly different training and inference settings. The results of 30 new experiments of inference encoders, each performed with 5-fold cross-validation are included in Figure 15. We compare `P-Fusion` and `MultiModN` on both tasks of the MIMIC dataset using all possible combinations of four input modalities at test time. `MultiModN` ignores missing modalities whereas `P-Fusion` imputes and therefore encodes missing modalities.

We note that the performance at inference for `P-Fusion` and `MultiModN` has no significant differences for all experiments (using 95% CIs). Figure 15 shows that, on average, `P-Fusion` tends to overfit more to the most dominant (visual) modality. When this modality is missing (at random or completely at random), `MultiModN` performs better on a combination of the remaining modalities (demo, text, time series). In the case of missing modalities, the observed effect in Figure 15 is weak – confidence intervals overlap. Considering the MNAR (missing not-at-random) scenario described in Sec. 6.4, the difference becomes significant.