HynetImpute: Missing Pattern Specialized Imputation via Hypernetwork - Supplementary Material

A Experimental Details

A.1 Hypernetwork Weight Initialization

The initialization of the neural network weights is known to play a critical role in determining the stability and convergence of training. This is particularly true for models like hypernetworks. As hypernetworks aim to generate parameters of other networks conditioned on specific inputs, this makes the model training challenging compared to standard neural networks. Using the initialization like the PyTorch default setting (Kaiming uniform scheme), often leads to exploding gradients or extremely high initial loss in the context of hypernetwork. The reason is that Kaiming initialization scales weights based on the layer size and activation functions, which is optimized for feedforward networks with fixed architectures, but not hypernetworks which dynamically generate parameters of other networks. For our implementation, we initialize the weights of the hypernetwork by sampling from a normal distribution $\mathcal{N}(0, 0.001^2)$, and obtain satisfactory performance.

A.2 Details of Synthetic Data Generation

To systematically evaluate our imputation method, we generate synthetic datasets in the experiment section that exhibit diverse missing patterns and feature relationships. The dataset is designed to simulate real-world complexities by introducing structured feature subsets with distinct functional transformations, ensuring a wide range of dependencies among variables.

A.2.1 Data Structure

Each dataset consists of n_{samples} instances, each with d_{input} features. These features are divided into k_{subsets} disjoint subsets, where each subset follows a distinct transformation. The number of features per subset is determined as $d_{\text{subset}} = d_{\text{input}}/k_{\text{subsets}}$, ensuring that every subset has an equal share of the total feature space.

A.2.2 Feature Transformations

To introduce meaningful structure into the synthetic dataset, each feature subset undergoes a transformation selected from a predefined set of functions:

- **Linear Transformation**: Features in this subset are generated as $X = B + \epsilon$, where B is a randomly sampled base variable and ϵ is Gaussian noise.
- **Quadratic Transformation**: Features follow a nonlinear quadratic dependency $X = B^2 + \epsilon$.
- **Sine Transformation**: A periodic dependency is introduced using $X = \sin(B) + \epsilon$.
- **Exponential Transformation**: Features grow exponentially with $X = \exp(B/5) + \epsilon$, simulating nonlinearly increasing trends.
- **Categorical Features**: These features are generated by assigning discrete category values from $\{0, 1, 2\}$ with added noise.

Each subset is assigned a transformation in a cyclic manner, ensuring diversity across the feature space.

A.2.3 Missingness Simulation

To simulate missingness, a binary mask $M \in \{0, 1\}^{n_{\text{samples}} \times d_{\text{input}}}$ is generated, where each feature is independently set as missing with probability p_{missing}. The observed data X_{missing} is then obtained by: $X_{\text{missing}} = X_{\text{full}} \odot M$, where X_{full} is the complete dataset and $\odot$ denotes element-wise multiplication.

A.2.4 Outputs

The generated dataset consists of:

- **Ground Truth Data** (X_{full}): The complete dataset without missing values.
- **Observed Data** (X_{missing}): The dataset with missing entries according to M.
- **Missingness Mask** (M): A binary mask indicating observed (1) and missing (0) entries.

These datasets provide a controlled environment for evaluating imputation methods, enabling an assessment of their ability to handle structured missingness patterns and diverse feature dependencies.

A.3 Dataset Statistics

Table 1 summarizes the statistics of the UCI datasets used in the experiments.

Table 1. Statistics of the UCI datasets.

Dataset	# of Samples	# of Features
iris	150	4
heart	303	13
mammo	961	5
concrete	1030	8
com_crime	1994	101
autism	704	12
ozone	5070	72
mice	1080	77

A.4 Ablation Study

Figure 1 shows the ablation study for all UCI datasets. HynetImpute achieves consistently better performance than removing missingness augmentation and hypernetwork.

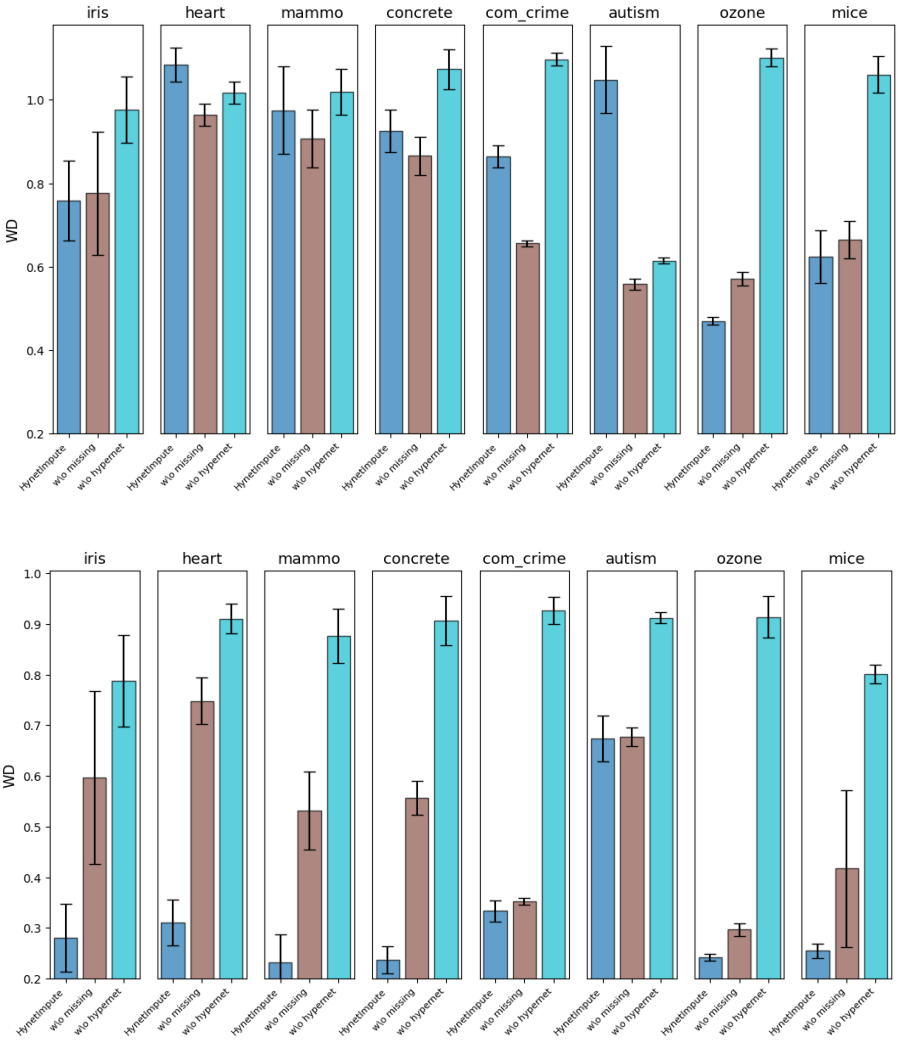

Figure 1. Ablation study based on UCI datasets.

A.5 Experiments under Different Missing Scenarios

We compared the imputation performance for different missing scenarios.

For the MCAR scenario, missing entries are introduced uniformly at random across the dataset. Each data value is independently masked, regardless of its value or any other features.

Under the MAR mechanism, missingness is introduced in a way that depends on other observed values (referring to Section 5.2).

For MNAR, we first simulate missingness as in the MAR mechanism above, using a subset of features as always-observed drivers and masking others via the logistic model. This provides an initial pattern where some features have missing values correlated with the (fully observed) drivers. Next, take the previously observed input features and mask a fraction of their values at random. In other words, those features that were kept complete for the MAR step are now themselves subjected to MCAR masking. This additional step means that the final dataset has missing values even in those driver features.

Figures 2-9 shows the performance comparison of the imputation results among the methods being tested based on UCI datasets with missingness under the MCAR, MAR and MNAR scenarios at the missingness ratio of 0.2, 0.5 and 0.8. Overall, HynetImpute performs better in MAR and MNAR settings, achieving the best or second-best in most settings.

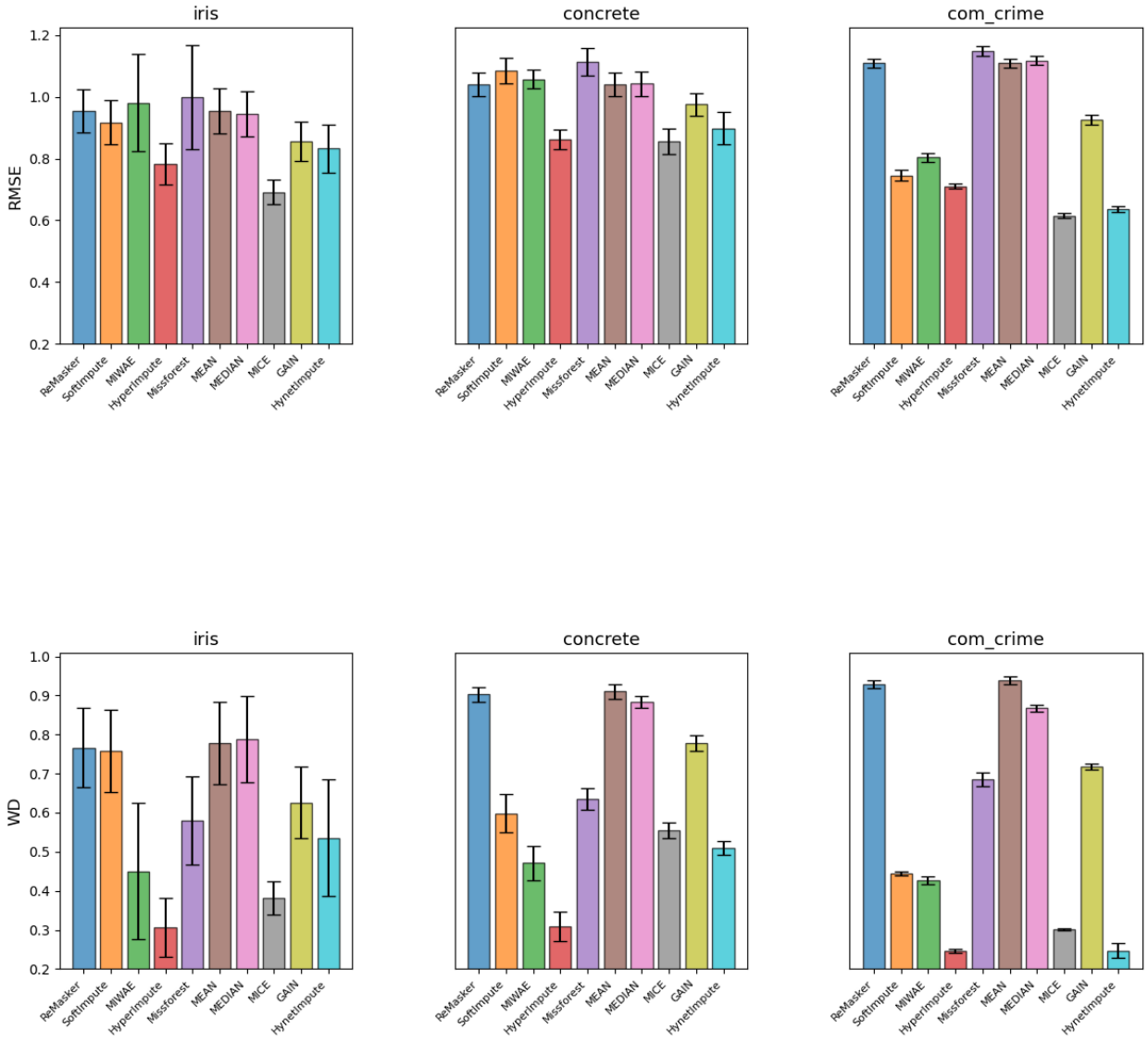

Figure 2. Performance comparison under missing completely at random (MCAR) with missing ratio 0.2.

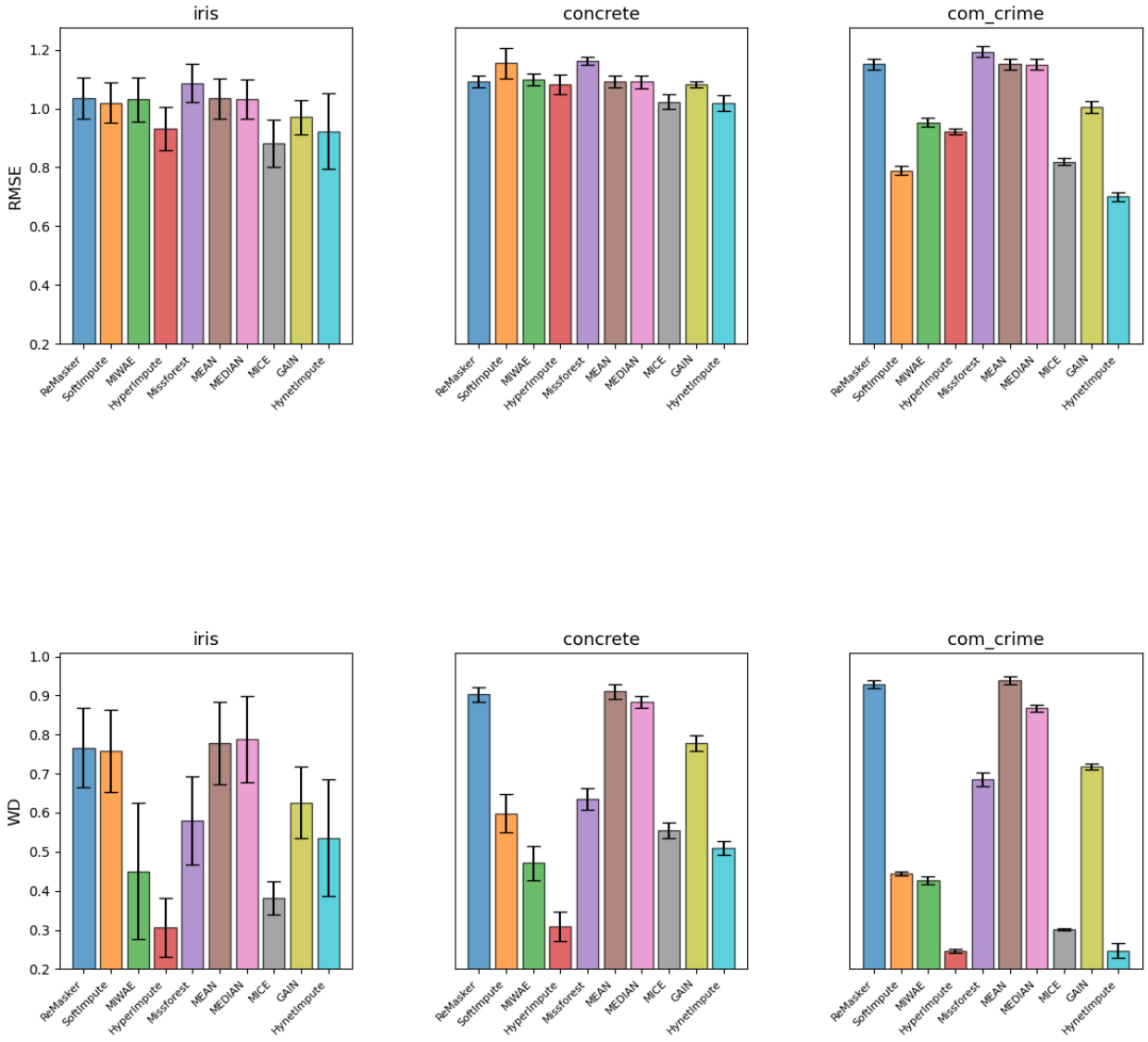

Figure 3. Performance comparison under missing completely at random (MCAR) with missing ratio 0.5.

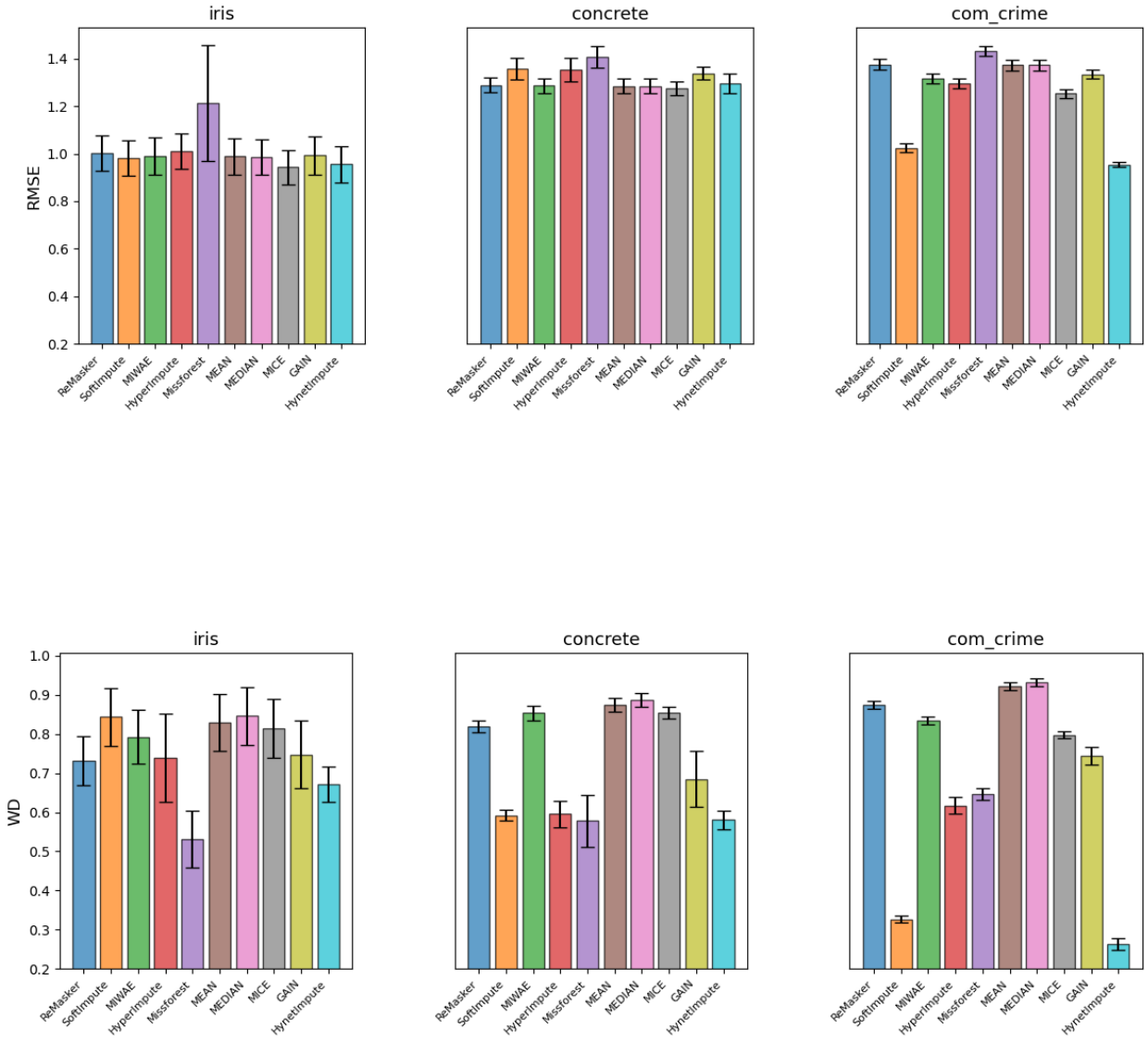

Figure 4. Performance comparison under missing completely at random (MCAR) with missing ratio 0.8.

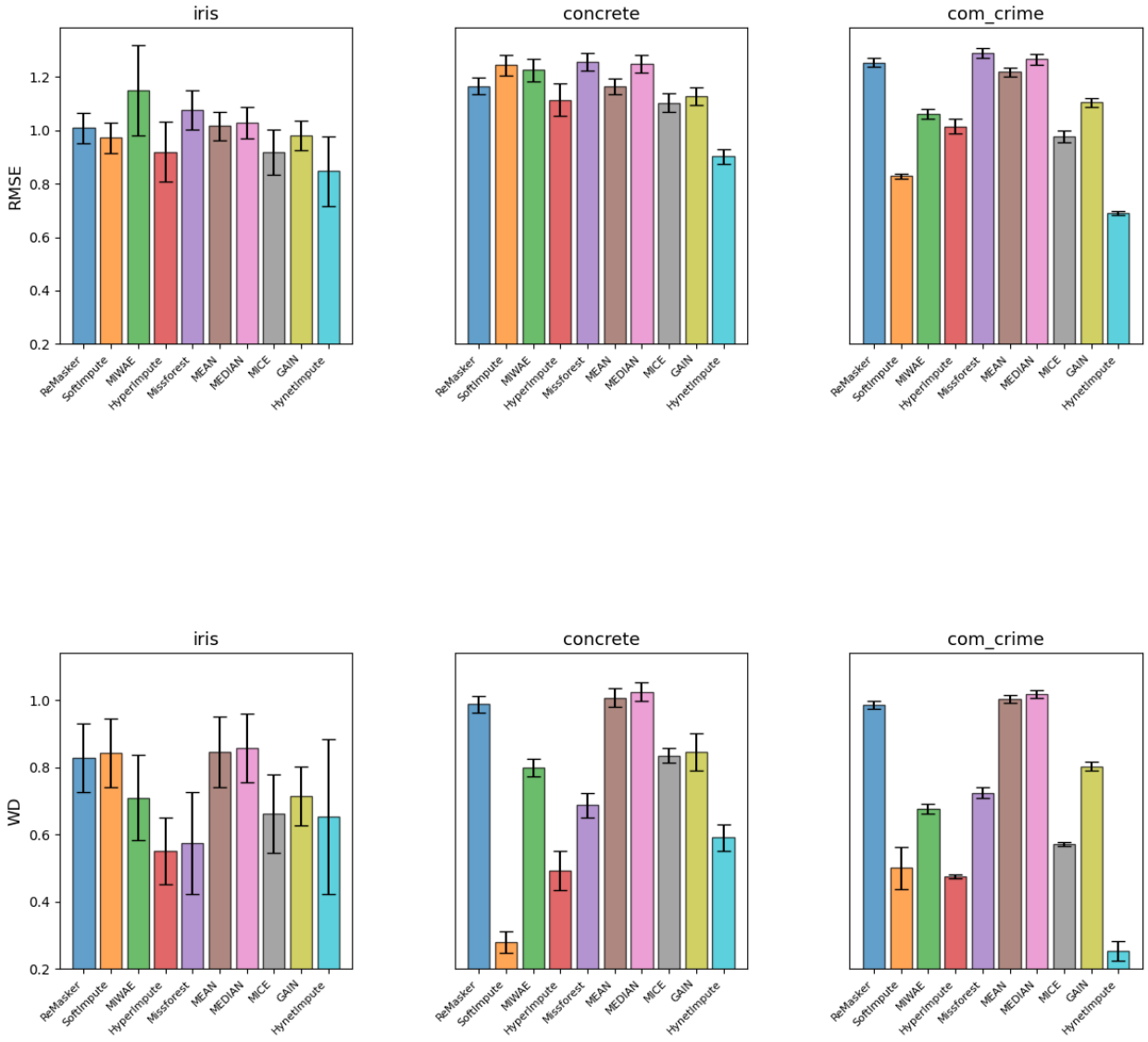

Figure 5. Performance comparison under missing at random (MAR) with missing ratio 0.5.

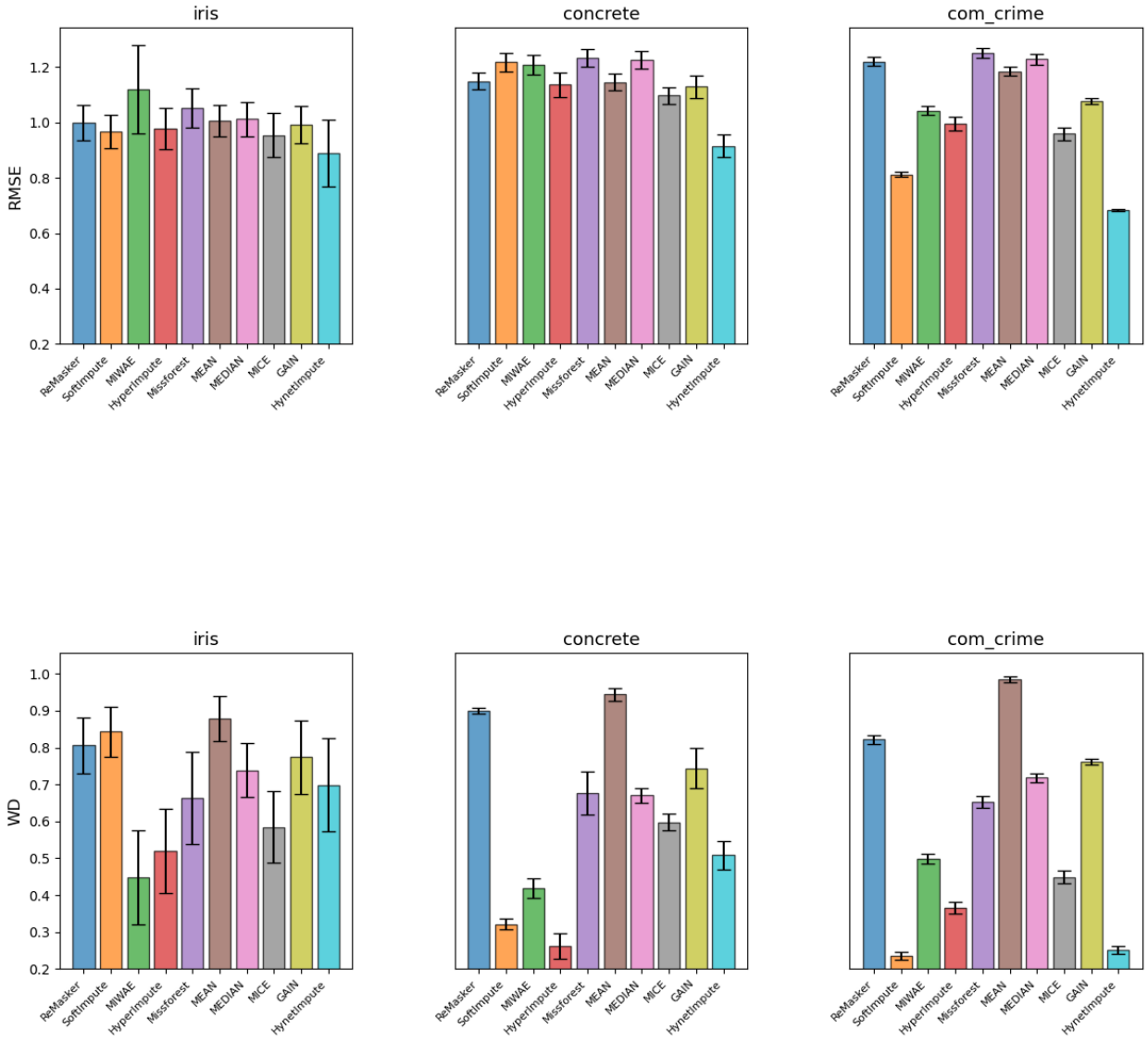

Figure 6. Performance comparison under missing at random (MAR) with missing ratio 0.8.

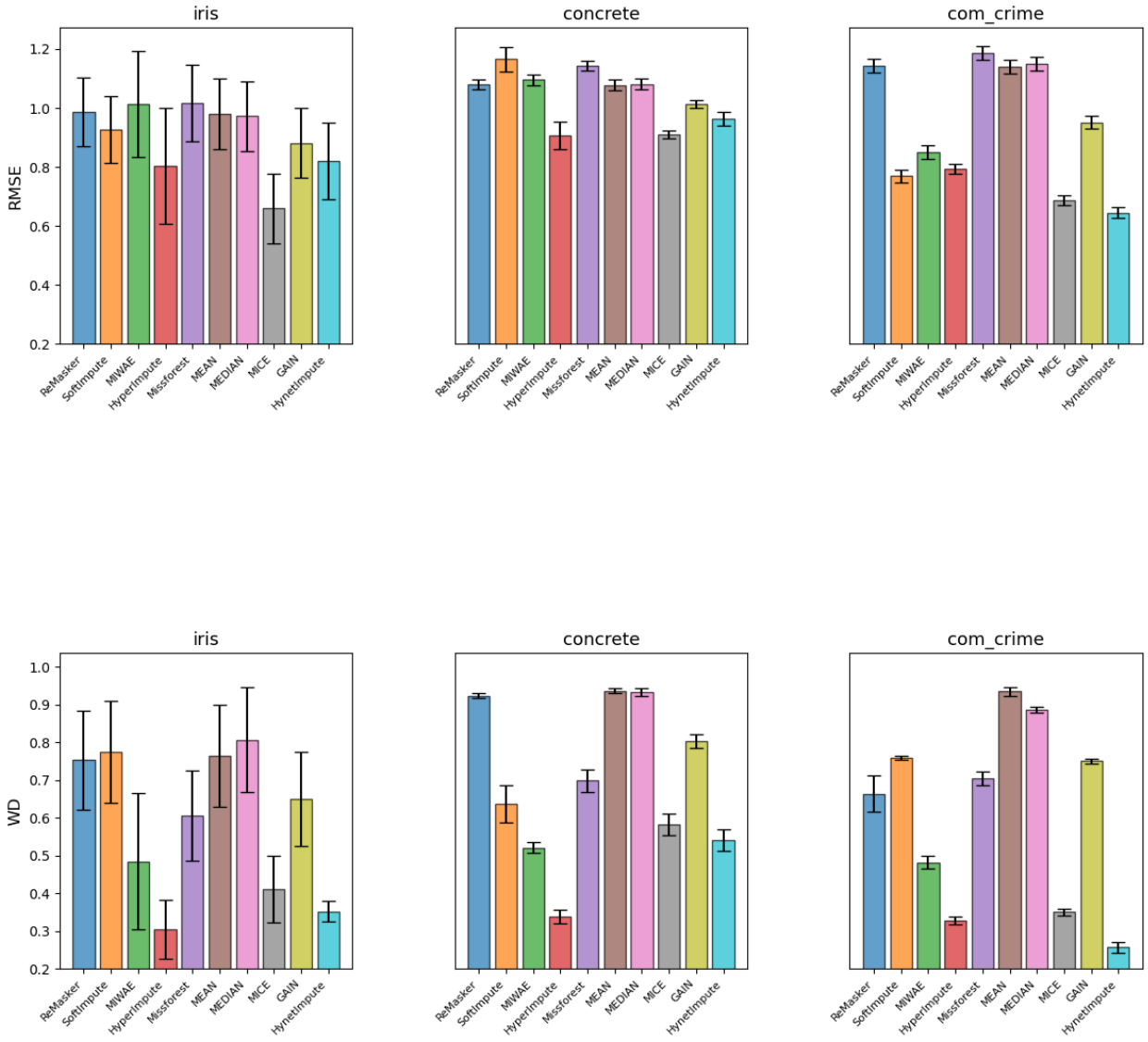

Figure 7. Performance comparison under missing not at random (MNAR) with missing ratio 0.2.

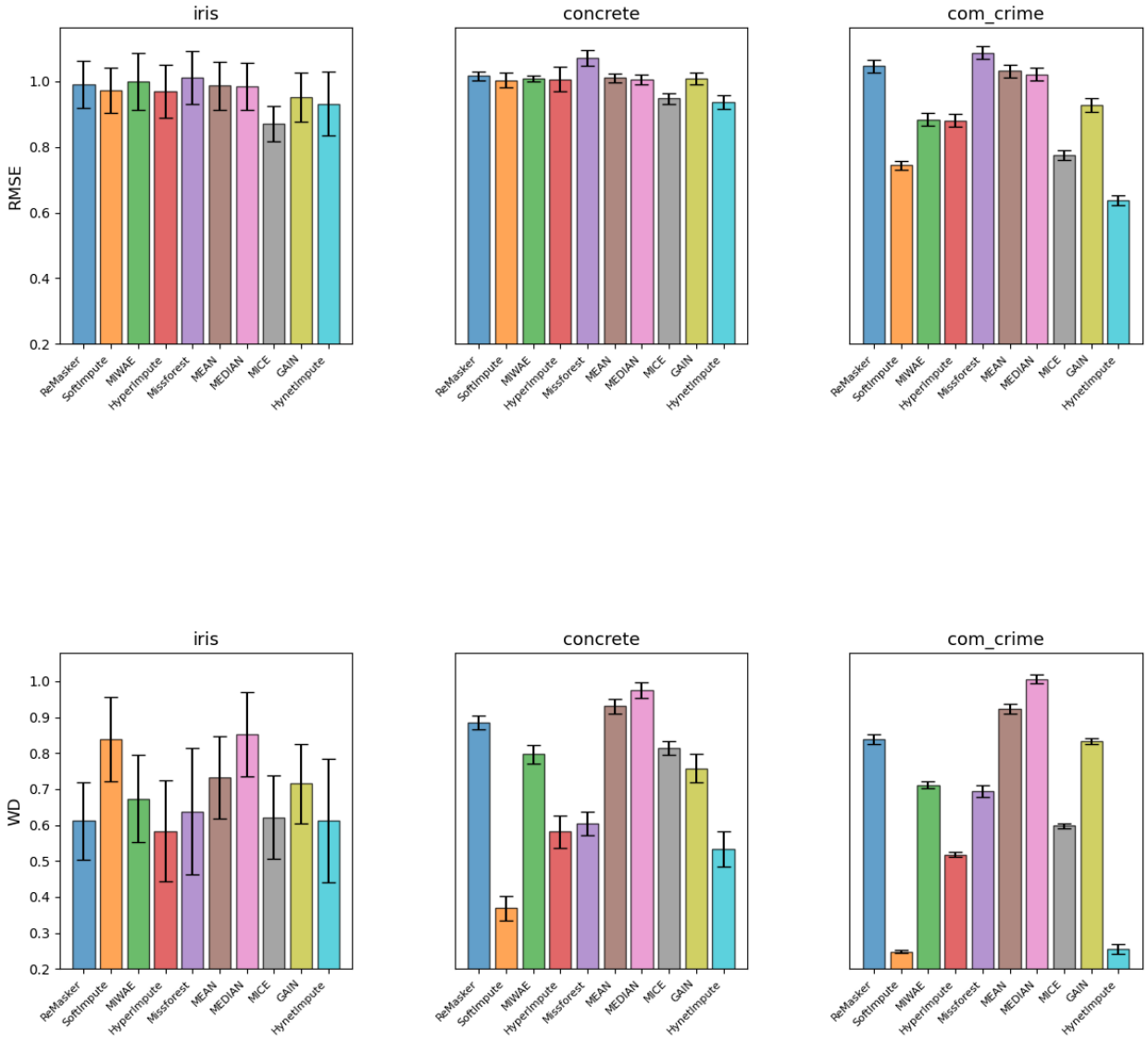

Figure 8. Performance comparison under missing not at random (MNAR) with missing ratio 0.5.

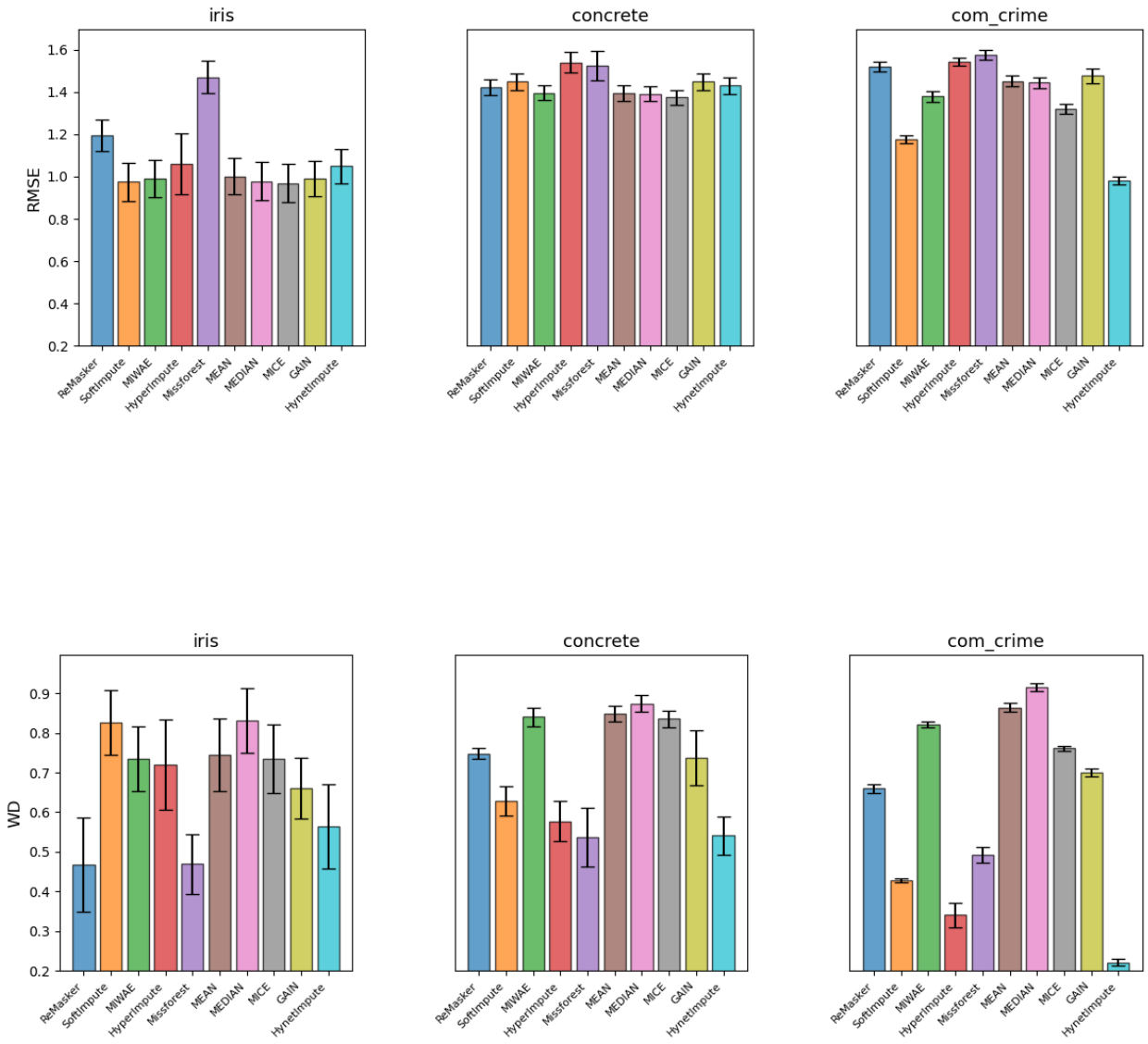

Figure 9. Performance comparison under missing not at random (MNAR) with missing ratio 0.8.