7. Appendix

7.1. Intuition Behind Gradient Consistency Analysis

In this section, we provide an intuitive explanation of our gradient consistency analysis and why it reveals the under-optimization of tail modality combination groups.

7.1.1. The Core Problem: Which Examples Drive Training?

When training a neural network with gradient descent, not all examples contribute equally to parameter updates. The actual update direction is determined by the aggregate gradient across the entire dataset. If certain subgroups (e.g., tail modality combinations) have gradient directions that diverge from this aggregate direction, their optimization objectives are effectively ignored during training.

7.1.2. From Individual Gradients to Dominant Direction

Given a dataset $\mathcal{X} = \{x_1, \ldots, x_n\}$ and model parameters θ, we can compute the gradient of each example's loss with respect to the parameters: $\nabla_\theta \ell(f(x_i;\theta), y_i)$. These gradients form the Jacobian matrix:

$$\mathbf{J}_\theta(\mathcal{X}) = \begin{bmatrix} \nabla_\theta f(x_1;\theta) \\ \nabla_\theta f(x_2;\theta) \\ \vdots \\ \nabla_\theta f(x_n;\theta) \end{bmatrix} \in \mathbb{R}^{n \times p}, \tag{9}$$

where p is the number of parameters. Each row represents how one example's output changes with respect to all parameters.

The Neural Tangent Kernel (NTK) matrix captures how these individual gradients interact:

$$\mathbf{\Theta}(\mathcal{X}, \mathcal{X}) = \mathbf{J}_\theta(\mathcal{X})\mathbf{J}_\theta(\mathcal{X})^\top \in \mathbb{R}^{n \times n}, \tag{10}$$

where $\Theta_{ij} = \langle \nabla_\theta f(x_i), \nabla_\theta f(x_j) \rangle$ measures the alignment between example i's and example j's gradients.

7.1.3. Extracting the Dominant Gradient Direction

The NTK matrix $\mathbf{\Theta}$ is symmetric and positive semi-definite, so we can perform eigendecomposition:

$$\mathbf{\Theta} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^\top, \tag{11}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ are eigenvalues and $\{\mathbf{u}_i\}$ are orthonormal eigenvectors.

The top eigenvector $\mathbf{u}_{\max} = \mathbf{u}_1$ (corresponding to the largest eigenvalue λ_1) captures the dominant direction in example-space—the linear combination of examples that collectively exerts the strongest influence on training dynamics. Projecting the Jacobian onto this direction gives us the aggregate parameter gradient:

$$\mathbf{g}_\theta(\mathcal{X}) = \mathbf{u}_{\max}^\top \mathbf{J}_\theta(\mathcal{X}) \in \mathbb{R}^p. \tag{12}$$

Intuition: $\mathbf{u}_{\max}$ identifies which weighted combination of examples dominates the gradient updates. If head groups (with many samples) align with $\mathbf{u}_{\max}$, they drive training. If tail groups diverge from $\mathbf{u}_{\max}$, they are under-optimized.

7.1.4. Measuring Gradient Consistency for Subgroups

For a modality combination group $g_k \subseteq \mathcal{X}$, we repeat the process:

1. Compute $\mathbf{J}_\theta(\mathcal{X}_{g_k})$ for samples in group g_k.
2. Form $\mathbf{\Theta}(\mathcal{X}_{g_k}, \mathcal{X}_{g_k}) = \mathbf{J}_\theta(\mathcal{X}_{g_k})\mathbf{J}_\theta(\mathcal{X}_{g_k})^\top$.
3. Extract the top eigenvector $\mathbf{u}'_{\max}$ and compute $\mathbf{g}_\theta(\mathcal{X}_{g_k}) = (\mathbf{u}'_{\max})^\top \mathbf{J}_\theta(\mathcal{X}_{g_k})$.

We then define the **gradient consistency score** as the cosine similarity between the group's dominant gradient direction and the overall dataset's dominant gradient direction:

$$\text{GC}_\theta(g_k) = \frac{\mathbf{g}_\theta(\mathcal{X}) \cdot \mathbf{g}_\theta(\mathcal{X}_{g_k})}{\|\mathbf{g}_\theta(\mathcal{X})\| \|\mathbf{g}_\theta(\mathcal{X}_{g_k})\|}. \tag{13}$$

Interpretation:

- $\text{GC}_\theta(g_k) \approx 1$: Group g_k's preferred update direction aligns with the overall training direction $\Rightarrow$ group is well-optimized.
- $\text{GC}_\theta(g_k) \ll 1$: Group g_k's preferred direction diverges from overall training $\Rightarrow$ group is under-optimized.

7.1.5. Why This Matters for Long-Tailed Modality Combinations

In our experiments (Figure 1(b)), we observe:

- **Head groups** (frequent modality combinations) maintain high gradient consistency throughout training because their large sample counts dominate the NTK's top eigenvector $\mathbf{u}_{\max}$.
- **Tail groups** (rare modality combinations) exhibit low and decreasing gradient consistency as training progresses, indicating their optimization objectives are increasingly ignored.

This gradient divergence explains the performance gap between head and tail groups and motivates our group DRO approach, which explicitly reweights groups to prevent tail groups from being overlooked during optimization.

7.1.6. Connection to Neural Tangent Kernel Theory

Recent work [5, 7] has shown that in the NTK regime (wide networks with small learning rates), the gradient descent dynamics are governed by the kernel's eigenspectrum. The top eigenvector $\mathbf{u}_{\max}$ determines which directions in function space are learned fastest. Our analysis leverages this insight to quantify when subgroups are aligned or misaligned with the dominant learning direction, providing a principled explanation for the long-tail performance gap.

Recap of key formulas

$$\Theta(X, X) = J_\theta(X)\, J_\theta(X)^T, \tag{14}$$

$$\Theta = U\, \Lambda\, U^T, \quad u_{\max} = \text{eigenvector of } \lambda_1, \tag{15}$$

$$g_\theta(X) = u_{\max}^T\, J_\theta(X), \tag{16}$$

$$\mathrm{GC}_\theta(X') = \frac{g_\theta(X) \cdot g_\theta(X')}{\|g_\theta(X)\|\, \|g_\theta(X')\|}. \tag{17}$$

7.2. Soft MoE Routing and Uncertainty in Routing

This section describes the core Soft Mixture-of-Experts (SoftMoE) routing pipeline: normalization, logits, probabilities, deep entropy analysis, uncertainty metrics (including KL divergence), and final expert mixing. Each step shows PyTorch code, tensor shapes, mathematical formulas, and intuition.

7.2.1. 1. Input & Shapes

- **Input embeddings:** $x \in \mathbb{R}^{B\times M\times D}$
  - B: batch size
  - M: sequence length (# tokens)
  - D: feature dimension

7.2.2. 2. Normalize Input & Router Weights

```
# Normalize token embeddings
x_norm   = F.normalize(x, dim=-1)
    # [B, M, D]
# phi: router params, shape [D, N, P]
phi_norm = self.scale * F.normalize(self.phi,
    dim=0)  # [D, N, P]
```

Math

$$\tilde{x}_{b,m,:} = \frac{x_{b,m,:}}{\|x_{b,m,:}\|_2},$$

$$\tilde{\phi}_{:,i,p} = s\, \frac{\phi_{:,i,p}}{\|\phi_{:,i,p}\|_2}.$$

Intuition Decouples magnitude from direction for stable routing.

7.2.3. 3. Compute Routing Logits

```
# Compatibility scores
logits = torch.einsum("bmd,dnp->bmnp", x_norm,
    phi_norm)  # [B, M, N, P]
```

Math

$$L_{b,m,i,p} = \sum_{d=1}^{D} \tilde{x}_{b,m,d}\, \tilde{\phi}_{d,i,p}.$$

Intuition Higher L means token m matches expert i slot p strongly.

7.2.4. 4. Convert to Probabilities (Softmax)

```
prob = torch.softmax(logits, dim=2)  # [B, M, N,
    P]
```

Math

$$p_{b,m,i,p} = \frac{\exp(L_{b,m,i,p})}{\sum_{j=1}^{N} \exp(L_{b,m,j,p})}.$$

7.2.5. 5. Deep Dive: Entropy as Uncertainty

Entropy quantifies the *spread* of the router's distribution over experts (and slots):

```
entropy = -(prob * torch.log(prob +
    1e-8)).sum(dim=2)  # [B, M, P]
```

Math

$$H_{b,m,p} = -\sum_{i=1}^{N} p_{b,m,i,p}\, \ln p_{b,m,i,p}, \quad 0 \le H \le \ln N.$$

$$\text{For } N = 32, \quad H_{\max} = \ln(32) \approx 3.47 \text{ nats.}$$

Interpretation

- $H \to 0$ (peaky): single expert dominates $\Rightarrow$ high certainty.
- $H \to \ln N$ (flat): uniform distribution $\Rightarrow$ high uncertainty.

7.2.6. 5.1 Entropy in Bits vs. Nats

- **Nats:** uses natural log, max $= \ln N$.
- **Bits:** use base-2 log, max $= \log_2 N$.

```
entropy_bits = -(prob * torch.log2(prob +
    1e-8)).sum(dim=2)
```

7.2.7. 5.2 Numerical Considerations

Add ϵ inside log to avoid $\log 0$, then:

- Clamp: `entropy = entropy.clamp_min(0).`
- Or: `torch.where(prob>0, torch.log(prob), 0).`

7.2.8. 6. Certainty & Uncertainty Metrics

We collect several metrics from $\text{prob} \in \mathbb{R}^{B\times M\times N\times P}$, all shape $[B, M, P]$:

Metric	Definition	Range
Normalized Certainty	$1 - H/\ln N$	[0,1]
Max Probability	$\max_i p_i$	[1/N,1]
Margin	$p_{(1)} - p_{(2)}$	[0,1-1/N]
Gini Impurity	$1 - \sum_i p_i^2$	[0,1-1/N]
Variance	$\sum_i (p_i - 1/N)^2$	[0,?]
KL vs Uniform	$D_{KL}(p\|u) = \sum_i p_i \ln \frac{p_i}{1/N}$	[0,ln N]

Table 6. Uncertainty metrics over router probabilities

```
# Max-prob
max_prob, _ = prob.max(dim=2)         # [B, M, P]
# Margin
sorted_p, _  = prob.sort(dim=2, descending=True)
margin       = sorted_p[...,0] - sorted_p[...,1]
# KL divergence from uniform
u       = 1.0 / self.num_experts
kl_div  = (prob * (torch.log(prob+1e-8) -
    torch.log(u))).sum(dim=2)
```

7.2.9. 7. Aggregating Over Dimensions

Aggregate $[B, M, P]$ to:

- Per-token: mean or max over slots P.
- Per-example: mean over tokens M.
- Global: mean over batch B.

```
cert_seq    = certainty.mean(dim=-1)       # [B]
avg_entropy = entropy.mean(dim=[1,2])      # [B]
```

7.2.10. 8. Final Expert Routing & Mixing

```
# Routing weights
d = torch.softmax(logits, dim=2)
        # [B, M, N, P]
# Combine weights
c = torch.softmax(logits.flatten(2),
    dim=-1).view_as(d)
# Expert-slot inputs
xs = torch.einsum("bmd,bmnp->bnpd", x_norm, d)
         # [B, N, P, D]
# Expert outputs
y_splits = [f(xs[:,i]) for i,f in
    enumerate(self.experts)]
ys       = torch.stack(y_splits, dim=1)
        # [B, N, P, D]
# Merge back
y = torch.einsum("bnpd,bmnp->bmd", ys, c)
         # [B, M, D]
```

Shapes Recap

Tensor	Shape
x_{norm}	$[B, M, D]$
ϕ_{norm}	$[D, N, P]$
logits	$[B, M, N, P]$
prob	$[B, M, N, P]$
entropy	$[B, M, P]$
certainty	$[B, M, P]$
d	$[B, M, N, P]$
xs	$[B, N, P, D]$
ys	$[B, N, P, D]$
c	$[B, M, N, P]$
y	$[B, M, D]$

7.2.11. Practical Insights on Entropy and KL Thresholds in Soft MoE

In practice, although our original paper employed an entropy-based threshold of $0.8 \times$ Max_Entropy, using a KL-divergence threshold of 0.1 yields similar performance. This equivalence arises because, without explicit regularization or expert specialization, the routing entropy naturally approaches the maximum entropy $\ln N$ for nearly all tokens, reflecting uniformly distributed routing probabilities across experts. Consequently, both a high entropy threshold and a low KL-divergence threshold effectively identify tokens exhibiting high uncertainty (near-uniform routing).

However, after regularization and expert specialization, the routing probabilities tend to cluster around a few experts, appearing visually sparse. Despite this apparent sparsity, the actual entropy value does not significantly decrease toward zero, due to the inherently soft nature of the routing probabilities in Soft MoE. This contrasts sharply with traditional sparse MoE systems.

To illustrate, consider a numerical example with $N = 32$ experts, where probabilities primarily cluster around five experts: $[0.2, 0.2, 0.15, 0.1, 0.05]$, totaling 0.7 probability mass, with the remaining 0.3 uniformly distributed among the other 27 experts. Computing the entropy and KL divergence from uniform for this distribution yields:

$$\begin{aligned} H &\approx 2.658 \text{ nats}, \\ H_{\max} &= \ln(32) \approx 3.466 \text{ nats}, \\ D_{\mathrm{KL}}(p\|u) &\approx 0.807. \end{aligned}$$

Even in this significantly specialized scenario, the entropy remains notably high (around 77 per cent of the maximum), clearly demonstrating that the entropy value in Soft MoE does not approach zero despite visual sparsity. Thresholding is good but not critically important in this paper, any reasonable value (depending on dataset and modalities) around 0.8-0.95 for Entropy and around 0.6-0.15 for KL empirically works for our use case.

7.3. Additional Results

As discussed in the main text, we ran 3 different splits (Random Seed 2, 42, 778 for FPRM, seed 0, 1, 42 for EMBED and MIMIC, these are chosen at random) and here are the Accuracy Standard Deviation, F1 Mean, and F1 Standard Deviation results. *Readers may notice there is a MC for FPRM with 100% accuracy and that is because this is a very small MC (0.2 (test)* 0.8% * 3394 samples). Results are consistent across 3 runs.*

7.4. Implementation Details

Code will be released regardless of acceptance after decision is out. Please search our paper's title by that time.

7.4.1. FPRM Implementation Details.

Dataset and Preprocessing. The FPRM dataset contains 3 394 examples from 1 683 patients, each with up to four

Modality				Multi-modal learning methods			Long tail				Ours
M1	M2	M3	M4	SoftMoE	FuseMoE	FlexMoE	FairMixup	Reweigh	GroupDRO	FairBatch	REMIND
1	0	0	0	0.772	<u>0.796</u>	0.770	0.635	0.760	**0.798**	0.729	0.752
1	1	1	0	0.870	<u>0.888</u>	0.871	0.708	0.590	0.799	0.858	**0.893**
1	0	0	1	0.671	0.758	0.756	0.559	**0.784**	0.749	0.648	<u>0.765</u>
1	1	0	1	0.521	0.521	0.711	0.587	0.521	<u>0.743</u>	0.587	**1.000**
1	1	1	1	0.671	<u>0.743</u>	0.729	0.545	0.497	0.687	0.657	**0.765**
Entire Dataset				0.742	<u>0.771</u>	0.754	0.626	0.607	0.736	0.709	**0.783**

Table 7. **Comparison of F1-score across modalities and methods on FPRM.** REMIND achieves the best.

Missing Scenarios				Multi-modal learning methods			Long Tail				Ours
M1	M2	M3	M4	SoftMoE	FuseMoE	FlexMoE	FairMixup	Reweigh	GroupDRO	FairBatch	REMIND
0	0	1	0	0.641	0.643	<u>0.646</u>	0.639	0.562	0.615	0.604	**0.658**
1	0	1	0	0.623	0.616	0.606	**0.697**	0.634	<u>0.676</u>	0.627	0.670
0	0	0	1	0.565	<u>0.614</u>	0.563	0.593	**0.618**	0.559	0.566	0.560
0	0	1	1	0.707	0.704	<u>0.709</u>	0.710	<u>0.720</u>	<u>0.720</u>	0.700	**0.751**
1	0	1	1	0.622	0.465	<u>0.648</u>	0.567	0.534	0.593	0.497	**0.673**
1	1	1	1	0.776	0.768	0.750	<u>0.780</u>	0.785	0.777	0.778	**0.803**
Entire Dataset				0.726	0.722	0.727	0.730	<u>0.736</u>	0.735	0.723	**0.758**

Table 8. **Comparison of F1-score across modalities and methods on EMBED.** REMIND achieves the best.

Missing Scenarios			Multi-modal learning methods			Long Tail				Ours
M1	M2	M3	SoftMoE	FuseMoE	FlexMoE	FairMixup	Reweigh	GroupDRO	FairBatch	REMIND
1	0	0	<u>0.625</u>	0.583	0.611	0.565	0.614	0.588	0.576	**0.645**
0	1	0	0.521	0.567	0.467	0.523	0.560	<u>0.568</u>	0.541	**0.573**
1	1	0	0.575	0.582	**0.615**	0.571	0.551	<u>0.591</u>	0.584	0.581
0	0	1	0.534	<u>0.589</u>	0.586	**0.608**	0.551	0.499	0.549	0.550
1	0	1	**0.630**	0.584	0.572	0.594	0.589	<u>0.627</u>	0.590	0.607
0	1	1	0.557	**0.598**	0.546	<u>0.577</u>	0.556	0.571	0.574	0.548
1	1	1	0.581	<u>0.601</u>	0.598	**0.603**	0.597	0.599	0.592	**0.603**
Entire Dataset			0.580	<u>0.591</u>	0.585	0.587	0.581	0.590	0.586	**0.594**

Table 9. **Comparison of F1-score across modalities and methods on MIMIC.** REMIND achieves the best.

modalities. All modalities goes through the PatchEmbedding Projection Layer to have the same final number of tokens (16) and embedding dimension (128).

- **M1:** 2D retinal fundus photographs, RGB, resized to 224×224, normalized to ImageNet mean/std.
- **M2:** 3D blood-flow volumes, 16 frames per sample; input permuted to (C, T, H, W) and trilinearly interpolated to 224×224 per frame.
- **M3:** 2D oxygen saturation maps, RGB, 224×224, normalized similarly to M1.

Modality				Multi-modal learning methods			Long tail				Ours
M1	M2	M3	M4	SoftMoE	FuseMoE	FlexMoE	FairMixup	Reweigh	GroupDRO	FairBatch	REMIND
1	0	0	0	0.063	0.067	0.076	0.170	0.041	0.068	0.145	0.069
1	1	1	0	0.127	0.130	0.043	0.077	0.371	0.072	0.124	0.028
1	0	0	1	0.141	0.025	0.014	0.139	0.030	0.062	0.134	0.012
1	1	0	1	0.247	0.247	0.342	0.361	0.441	0.290	0.361	0.000
1	1	1	1	0.235	0.123	0.116	0.173	0.221	0.200	0.171	0.185
Entire Dataset				0.097	0.087	0.065	0.056	0.167	0.110	0.093	0.094

Table 10. **Standard Deviation of F1-score across modalities and methods on FPRM.** REMIND achieves the best.

Missing Scenarios				Multi-modal learning methods			Long Tail				Ours
M1	M2	M3	M4	SoftMoE	FuseMoE	FlexMoE	FairMixup	Reweigh	GroupDRO	FairBatch	REMIND
0	0	1	0	0.015	0.021	0.028	0.009	0.032	0.028	0.031	0.018
1	0	1	0	0.034	0.088	0.055	0.031	0.037	0.137	0.095	0.042
0	0	0	1	0.015	0.036	0.044	0.124	0.114	0.018	0.105	0.073
0	0	1	1	0.003	0.002	0.004	0.021	0.006	0.006	0.021	0.020
1	0	1	1	0.006	0.052	0.156	0.124	0.105	0.066	0.046	0.112
1	1	1	1	0.003	0.004	0.040	0.010	0.010	0.005	0.020	0.004
Entire Dataset				0.004	0.002	0.003	0.014	0.006	0.004	0.016	0.003

Table 11. **Standard Deviation of F1-score across modalities and methods on EMBED.** REMIND achieves the best.

Missing Scenarios			Multi-modal learning methods			Long Tail				Ours
M1	M2	M3	SoftMoE	FuseMoE	FlexMoE	FairMixup	Reweigh	GroupDRO	FairBatch	REMIND
1	0	0	0.004	0.022	0.010	0.024	0.008	0.023	0.032	0.009
0	1	0	0.004	0.014	0.011	0.020	0.005	0.026	0.027	0.008
1	1	0	0.006	0.024	0.014	0.024	0.006	0.028	0.032	0.009
0	0	1	0.007	0.014	0.010	0.025	0.006	0.018	0.023	0.008
1	0	1	0.010	0.019	0.015	0.022	0.003	0.032	0.034	0.009
0	1	1	0.006	0.017	0.016	0.025	0.008	0.025	0.033	0.008
1	1	1	0.002	0.022	0.018	0.023	0.007	0.030	0.024	0.009
Entire Dataset			0.003	0.013	0.011	0.021	0.005	0.021	0.019	0.008

Table 12. **Standard Deviation of F1-score across modalities and methods on MIMIC.** REMIND achieves the best.

- **M4:** Tabular metadata are standardized to zero mean and unit variance, then split into 16 patches of size $\lceil 64/16 \rceil = 4$ for embedding using the patch embedding layer.

Modality-combination distribution:

MC1 (M1+M4)	59.9%
MC2 (M1 only)	17.7%
MC3 (M1+M2+M3+M4)	16.2%
MC4 (M1+M2+M3)	5.3%
MC5 (M1+M3+M4)	0.8%

Missing Scenarios				Multi-modal learning methods			Long Tail				Ours
M1	M2	M3	M4	SoftMoE	FuseMoE	FlexMoE	FairMixup	Reweigh	GroupDRO	FairBatch	REMIND
0	0	1	0	0.008	0.005	0.024	0.026	0.008	0.017	0.023	0.010
1	0	1	0	0.022	0.043	0.041	0.006	0.027	0.043	0.011	0.031
0	0	0	1	0.009	0.012	0.023	0.079	0.056	0.029	0.032	0.020
0	0	1	1	0.002	0.004	0.002	0.006	0.002	0.005	0.007	0.003
1	0	1	1	0.010	0.032	0.066	0.052	0.031	0.010	0.028	0.042
1	1	1	1	0.002	0.003	0.004	0.005	0.012	0.007	0.006	0.003
Total				0.002	0.003	0.002	0.007	0.005	0.006	0.009	0.003

Table 13. **Standard Deviation of ACC across modalities and methods on EMBED.** REMIND achieves the best.

Modality				Multi-modal learning methods			Long tail				Ours
M1	M2	M3	M4	SoftMoE	FuseMoE	FlexMoE	FairMixup	Reweigh	GroupDRO	FairBatch	REMIND
1				0.065	0.035	0.021	0.038	0.039	0.064	0.038	0.038
1	1	1		0.006	0.071	0.152	0.043	0.122	0.072	0.022	0.046
1			1	0.019	0.014	0.023	0.031	0.020	0.012	0.012	0.003
1	1		1	0.144	0.144	0.250	0.250	0.382	0.144	0.250	0.000
1	1	1	1	0.051	0.069	0.028	0.071	0.074	0.156	0.045	0.080
Overall				0.028	0.018	0.017	0.017	0.025	0.014	0.013	0.023

Table 14. **Standard Deviation of ACC across modalities and methods on FPRM.** REMIND achieves the best.

Missing Scenarios			Multi-modal learning methods			Long Tail				Ours
M1	M2	M3	SoftMoE	FuseMoE	FlexMoE	FairMixup	Reweigh	GroupDRO	FairBatch	REMIND
1	0	0	0.0161	0.0367	0.0185	0.0347	0.0129	0.0361	0.0617	0.0153
0	1	0	0.0140	0.0428	0.0231	0.0426	0.0061	0.0312	0.0242	0.0133
1	1	0	0.0104	0.0394	0.0217	0.0425	0.0157	0.0518	0.0434	0.0147
0	0	1	0.0095	0.0271	0.0208	0.0254	0.0069	0.0382	0.0357	0.0127
1	0	1	0.0106	0.0348	0.0212	0.0426	0.0069	0.0485	0.0495	0.0139
0	1	1	0.0132	0.0384	0.0251	0.0294	0.0079	0.0382	0.0286	0.0139
1	1	1	0.0141	0.0214	0.0287	0.0381	0.0059	0.0451	0.0482	0.0147
Total			0.0067	0.0189	0.0171	0.0346	0.0017	0.0349	0.0353	0.0136

Table 15. **Standard Deviation of ACC across modalities and methods on MIMIC.** REMIND achieves the best.

Model Architecture. All experiments use a REMIND MoE transformer with 32 experts and one slot per expert, embedding size 128, num of heads 8. Modality-specific encoders:

- **PatchEmbeddings** (M4): Linear(D→128) with dropout 0.25, *e.g.* projecting $1 \times 64 \rightarrow 16 \times 128$.
- **ResNet2DAdapter** (M1, M3): Pretrained ResNet-50, remove FC head, project 2048→512 via Kaiming-initialized fc.
- **ResNet3DAdapter** (M2): torchvision r3d_18 (Kinetics-

400), remove head, adaptive avg-pool, project 512→512.

Training Configuration.

- **Epochs:** 50; **Batch size:** 32 per GPU.
- **Optimizer:** AdamW with $\mathrm{lr} = 5 \times 10^{-5}$, $\mathrm{wd} = \mathrm{lr}/10$.
- **Runs:** 3 seeds, reporting mean $\pm$ std.

7.4.2. MIMIC Implementation Details.

Dataset and Preprocessing. We leverage the MIMIC dataset following [36, 42], using the clinic text, ICD-9 Code, and Lab test modalities, each sample has up to three modalities.

- **M1:** clinical text.
- **M2:** labs and vital values.
- **M3:** ICD-9 codes.

During preprocessing, we randomly masked some existing modalities for each record to construct comprehensive modality combinations, and the modality-combination's distribution is:

MC1 (M1 only)	4.8%
MC2 (M2 only)	5.2%
MC3 (M1+M2)	14.3%
MC4 (M3 only)	5.2%
MC5 (M1+M3)	15.5%
MC6 (M2+M3)	14.7%
MC7 (M1+M2+M3)	40.3%

Model Architecture. All experiments use a REMIND MoE transformer with 128 experts and one slot per expert, embedding size 768, num of heads 8. We treat each modality as text input, which are then encoded by modality-specific T5-base encoders [28].

Training Configuration.

- **Epochs:** Max 20; **Batch size:** 32 per GPU.
- **Optimizer:** AdamW with $\mathrm{lr} = 1 \times 10^{-4}$, $\mathrm{wd} = 5 \times 10^{-5}$.
- **Runs:** 3 seeds, reporting mean $\pm$ std.

EMBED Implementation Details.

Dataset and Preprocessing. We leverage the EMBED dataset following [39], and each sample has up to four modalities.

- **M1:** C-View synthetic Cranio-Caudal view image, resized to 224×224, normalized to ImageNet mean/std.
- **M2:** C-View synthetic Medio-Lateral Oblique view image, resized to 224×224, normalized to ImageNet mean/std.
- **M3:** full-field digital mammography Cranio-Caudal view image, resized to 224×224, normalized to ImageNet mean/std.
- **M4:** full-field digital mammography Medio-Lateral Oblique view image, resized to 224×224, normalized to ImageNet mean/std.

Modality-combination distribution:

MC1 (M3 only)	3.0%
MC2 (M1+M3)	0.6%
MC3 (M4 only)	1.2%
MC4 (M3+M4)	57.8%
MC5 (M1+M3+M4)	0.3%
MC6 (M1+M2+M3+M4)	37.1%

Model Architecture. All experiments use a REMIND MoE transformer with 128 experts and one slot per expert, embedding size 768, num of heads 8. Each modality are firstly projected by modality-specific ViT-base encoders [6].

Training Configuration.

- **Epochs:** Max 20; **Batch size:** For FairMixup, 16 per GPU because it takes larger GPU memory; For others, 32 per GPU.
- **Optimizer:** AdamW with $\mathrm{lr} = 2 \times 10^{-5}$ for FairMixup and $\mathrm{lr} = 5 \times 10^{-5}$ for others, $\mathrm{wd} = 5 \times 10^{-5}$.
- **Runs:** 3 seeds, reporting mean $\pm$std.

Main experiments are conducted on NVIDIA A100-SXM4-80GB GPUs. All datasets use FocalLoss(γ=2), and a scheduler of StepLR($\mathrm{step\_size} = 5$, $\gamma = 0.1$).

7.5. Discussion Q3/Table 5 Experiment Explained

Extreme Missingness Robustness Analysis. To evaluate robustness under extreme missing scenarios, we systematically simulate high missingness rates for individual modalities. For each modality M_i (where $i \in 1, 2, 3, 4$), we artificially remove it from 80% of samples, creating two distinct groups: the "Non-Missing" group containing the remaining 20% of samples that retain M_i, and the "Missing" group comprising the 80% of samples without M_i. We then evaluate both baseline methods and our approach on: (1) overall performance across all samples, (2) performance on the Non-Missing group, and (3) performance on the Missing group. This design allows us to assess whether methods can effectively utilize sparse modalities when they are available while maintaining robust performance when they are absent. The results in Table 5 demonstrate that our method exhibits superior robustness compared to baselines. Each row represents the experiment where modality M_i is made 80% missing. Notably, our approach maintains more consistent performance across both groups, with smaller performance gaps between Non-Missing and Missing groups, indicating better adaptation to varying modality availability patterns.

7.6. Hyperparameter Sensitivity on EMBED

We study the sensitivity of REMIND to several key hyperparameters on the EMBED dataset under different missing-modality patterns. Table 16 reports results for six representative missingness scenarios (rows S1–S6), where the binary block on the left indicates which of the four modalities

Scenario	Modality presence (1 = observed)				DRO step size η		# fusion experts E			Initialization	
	M1	M2	M3	M4	0.1	0.5	32	64	128	zero emb.	sinus-resid.
S1			1		74.6	75.6	73.8	74.1	74.6	73.3	73.6
S2	1		1		81.7	77.4	82.8	81.7	79.9	84.9	77.4
S3				1	75.7	75.1	70.5	78.0	74.6	75.1	76.9
S4			1	1	79.2	79.1	78.8	79.2	79.5	79.1	79.3
S5	1		1	1	74.5	76.4	72.7	69.1	74.0	69.1	65.5
S6	1	1	1	1	84.0	84.0	84.0	84.4	84.0	84.1	83.5
Avg					80.5	80.4	80.2	80.6	80.7	80.5	80.3

Table 16. **Hyperparameter sensitivity of REMIND on the EMBED dataset across six representative missingness scenarios.** Columns on the left indicate which modalities (M1–M4) are observed in each scenario (1 = present, blank = missing). We vary the group-DRO step size η, the number of fusion experts E, and initialization schemes for imputation embeddings and group-specific components. The last row reports the average performance (macro AUROC) over all scenarios.

(M1–M4) are observed in each scenario (1 = present, blank = missing). For each scenario we vary:

- the group-DRO step size $\eta \in \{0.1, 0.5\}$,
- the number of fusion experts $E \in \{32, 64, 128\}$,
- imputation embedding and initialization schemes for the group-specific router (disabling learnable imputation embeddings, "zero emb", and sinusoidal initialization of residual routing matrices, "sinus-resid").

Across scenarios, the average accuracy (last row) remains tightly clustered between 80.2% and 80.7%, indicating that REMIND is fairly insensitive to these hyperparameters. Increasing the number of experts from 32 to 64 or 128 yields small but consistent gains (from 80.2% to 80.6% and 80.7%), while changing the DRO step size from $\eta = 0.1$ to $\eta = 0.5$ has negligible effect (from 80.5% to 80.4% on average). The routing initialization choices and whether to use a learnable imputation embedding and also lead to similar overall performance (80.3% and 80.5%).

Method	Parameters	Step per Sec	Flops
SoftMoE	958 M	0.41	146.7 G
REMIND	960 M	0.42	146.7 G
REMIND (with group-specific routing)	960 M	0.68	146.9 G

Table 17. **Computation costs.**

REMIND uses an exponentiated-gradient update for the group weights $\{\lambda_k\}$ in the group-DRO objective, with a sharpness parameter γ controlling how aggressively the optimizer focuses on high-loss groups. On the EMBED dataset, we sweep $\gamma \in \{0.5, 0.1, 0.02\}$ and obtain overall accuracies of 80.4%, 80.5%, and 80.7%, respectively. The best performance is achieved with the smallest value $\gamma = 0.02$ (used in our main experiments), suggesting that a milder sharpness that does not over-concentrate on the worst groups yields slightly better overall robustness, while performance remains stable across this range.

7.7. Computation Cost

Here we report the computation cost of REMIND and a MoE baseline method. The two methods are implemented with the same encoder, and the same configuration of the expert layer in the Soft MoE block. The only difference is that REMIND is built on distributionally robust learning framework, and has modality-specific routing matrices in the second stage training. We compare the parameter size, training time (iteration per second), and the FLOPs per sample in Table 17. The training and inference time is per step, where the batch size is 32.

7.8. Limitations

Our study has several limitations that suggest promising directions for future work. Our empirical evaluation focuses on clinical datasets with three to four modalities constrained by the availability of public high-modality benchmarks, while our proposed framework is generalizable and scalable to even more modalities. REMIND only adds lightweight group-specific residual routing and embedding parameters based on a shared expert pool with small computational overhead. We also primarily investigate medical classification tasks and extension to segmentation and other multi-task learning is a promising extension. We hope that REMIND provide a useful foundation for building robust and scalable multi-modal systems under realistic high-modality missingness.