Team Template Submission

Instructions

Please complete the following tables with information relevant to your team and submission. Follow the instructions provided for each section carefully to ensure compliance with submission guidelines.

1 Team and Participant Information

Instructions: Provide details about your team. Teams ranked in the top 6 can list up to 5 contributors; all other teams may list up to 3 contributors.

Team Name	Contributor Names (max. 5/3)	Institution	GitHub Link
YouvenZ	Rachid Zeghlache[1], Ikram Brahim[1,2], Gwenolé Quellec [1]	LaTIM UMR 1101[1], CHU Brest[2]	https://github.com/YouvenZ/MARIO-Challenge-MICCAI-2024

2 Task 1: Method Summary

Instructions: Provide a summary of the methods used for Task 1, including preprocessing, model architecture, and other relevant information.

Preprocessing	Encoder	Loss Function	Post-processing	Data Augmentation	Inference Time (GPU)	Model Params	Framework
crop, resize, normalization	ResNet50	BCE	Ensembling	Rotation, Zoom	7m 6s	100M	PyTorch

3 Task 1: Methodology Components

Instructions: Indicate whether specific components were used in Task 1. For each "Yes" answer, provide the name of the component, in case you did not use the component respond by "N".

Pretext Task (Y/N + Name)	Foundation Model (Y/N + Name)	Multi-modal Learning (Y/N + Name)	Public Dataset (Y/N + Name)
Y / Masked Autoencoder	Y / CLIP	Y / Cross-attention Fusion	N

4 Task 2: Method Summary

Instructions: Provide a summary of the methods used for Task 2, including preprocessing, model architecture, and other relevant information.

Preprocessing	Encoder	Loss Function	Post-processing	Data Augmentation	Inference Time (GPU)	Model Params	Framework
resize, histogram matching	DenseNet121	Focal Loss	Majority Voting	Brightness Shift	5m 46s	50M	Keras

5 Task 2: Methodology Components

Instructions: Indicate whether specific components were used in Task 2. For each "Yes" answer, provide the name of the component, in case you did not use the component respond by "N".

Pretext Task (Y/N + Name)	Foundation Model (Y/N + Name)	Multi-modal Learning (Y/N + Name)	Public Dataset (Y/N + Name)
Y / SimCLR	N	Y / Multi-modal Fusion	N

6 Team Approach Summaries

(Methodology, architecture details, preprocessing, and data augmentation strategies, and other specific modules developed.)

6.1 YouvenZ - Example Team Summary

Task 1:

The YouvenZ team designed a robust neural network model, RetinaTrackNet, tailored for identifying progression changes in retinal OCT scans over time. The preprocessing pipeline included intensity normalization, resizing each B-scan to 256x256 pixels, and anatomical alignment across scans to ensure consistency. To counter class imbalance, they employed SMOTE (Synthetic Minority Over-sampling Technique) on the training set and supplemented it with targeted data augmentation, applying transformations such as RandomRotation, GaussianNoise, and RandomRescale, aiming to generate realistic variations in retinal structures. RetinaTrackNet's architecture was based on a ResNet-50 backbone integrated with a temporal convolutional network (TCN) layer to capture sequential changes between scan pairs. Additionally, a self-attention mechanism was applied to highlight critical areas in each scan, improving model focus on disease-relevant features. The training procedure involved fine-tuning a pre-trained ResNet-50 model with an Adam optimizer and a cyclic learning rate schedule to maximize convergence stability. Post-processing involved a probabilistic thresholding method, assigning higher confidence scores to high-probability predictions, which helped reduce false positives. The team noted strong performance with an increased F1-score, attributing this to the self-attention layer and dynamic data augmentation.

Task 2:

For Task 2, YouvenZ introduced ProgressionForecaster, a hybrid model aimed at predicting AMD progression within a three-month window. Preprocessing steps were similar to Task 1, but additional care was taken to ensure data consistency by performing histogram matching across datasets from different devices. This task also required addressing a significant class imbalance; the team tackled this with class-weighted focal loss during training, which emphasized the minority classes. ProgressionForecaster's architecture consisted of a DenseNet-121 backbone coupled with a Long Short-Term Memory (LSTM) network to model temporal dependencies. To handle the ordinal nature of the progression labels, the team utilized a custom loss function based on the Earth Mover's Distance (EMD), which helped capture the gradation between classes. Data augmentation involved transformations like brightness shifts, GaussianBlur, and RandomAffine, applied specifically to minority classes to boost model robustness without overfitting. Post-processing incorporated a majority voting scheme across the B-scans within each OCT volume, ensuring volume-level consistency in predictions. The training process used a mix of the AdamW optimizer and cosine annealing for learning rate decay, with dropout layers incorporated within the LSTM to improve generalization. This approach achieved promising results, particularly for minority class predictions, although the team highlighted the need for further refinement in multi-class predictions. The code for the team's solution is available at https://github.com/YouvenZ/MARIO-Challenge-MICCAI-2024.