3NETLLM DESIGN

NetLLM is designed with efficient APIs and modules that can be easily integrated into existing SL and RL codebases, significantly reducing the cost of adapting LLMs to network tasks while retaining its robust pre-trained knowledge.

3.1 API Design

NetLLM is implemented using Python and Bash, and it uses three APIs: Adapt, Test, and RL\_Collect to interact with the existing codebase.

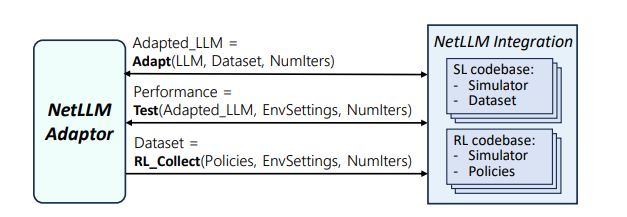


Figure 9: Components and interfaces needed to integrate NetLLM with an existing SL/RL codebase for LLM adaptation.

3.1.1Adapt

Use the provided dataset to adaptively tune the LLM, learn domain knowledge for a specific network task, and return an adjusted LLM snapshot.

3.1.2Test

Evaluate the performance of the adapted LLM in a test environment for a given simulation setup.

3.1.3RL\_Collect

For RL tasks that do not have an off-the-shelf dataset, interact with the environment using a given RL policy to collect empirical datasets for subsequent adaptation to the LLM.

3.2 Module implementation

Through the design of multimodal encoders, network headers, and low-rank matrices, NetLLM can efficiently process multimodal data and generate valid answers for specific network tasks.

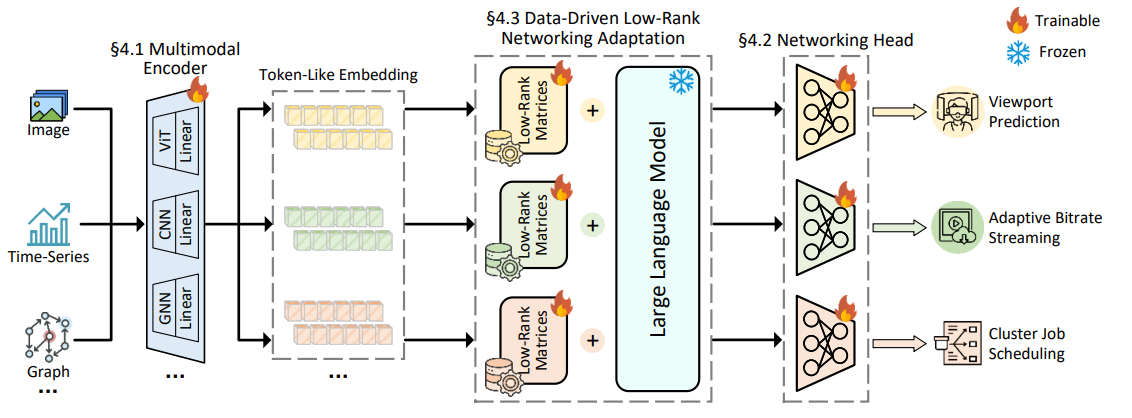


Figure 5: NetLLM consists of three core components: multimodal encoder to encode task inputs, networking head to generate task-specific answers and data-driven low-rank networking adaptation to efficiently learn domain knowledge for networking. The framework is illustrated with three tasks: VP, ABR and CJS, but all ideas can be easily applied to other networking tasks.

3.2.1 Multimodal Encoder

The core purpose of a Multimodal encoder is to process non-textual information. Feature encoder is used to extract features using some existing high-performance feature encoders for different kinds of features. Linear projection is used to solve the problem that the extracted feature dimensions are not equal to the required input dimensions. The multimodal encoder encodes the multimodal input data in the network task into token-like embeddings that can be processed by the LLM. For different modalities of data, specific feature encoders are used to extract features, such as Vision Transformer (ViT) to extract image features; using 1D-CNN (One-Dimensional Convolutional Neural Network) to extract features from time series data; Extract features of scalar data using a fully connected layer and feature extracting graph data using graph neural networks (GNNs).

Finally, the extracted features are projected into the embedding space consistent with the input dimension of the LLM through a linear layer, and Layer Normalization is used to ensure the stability of the training process.

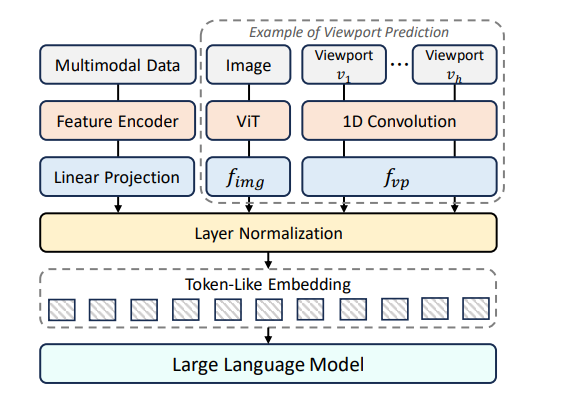


Figure 6: Illustration of the multimodal encoder of NetLLM to encode multimodal data.

3.2.2 Networking Head

The network header designs a lightweight trainable projector for each network task, mapping the output features of the LLM directly to the task-specific answer space. Generate answers directly from the output of the LLM instead of word-by-word predictions, reducing generation delays and ensuring the validity of answers. The core role of the network header is to output non-text content.

3.2.3 Data-Driven Low-Rank Networking Adaptation (DD-LRNA)

DD-LRNA optimizes the model by using a static, pre-collected dataset, improving the efficiency of training and reducing the complexity and resource consumption of real-time data processing.

For prediction tasks, the LLM was fine-tuned using a standard supervised learning (SL) data-driven training process. At the same time, for decision-making tasks, data-driven reinforcement learning (RL) technology is used to fine-tune the LLM through the empirical dataset collected by the interaction between existing algorithms and the environment, so as to avoid direct interaction with the environment.

In order to retain the pre-training knowledge of LLMs, all parameters of LLMs are frozen so that they remain unchanged during the fine-tuning process, and low-rank matrices with dimensions much smaller than the original matrices are introduced, so as to significantly reduce the training cost. At each training step, a random batch of data is sampled from the dataset, fed into the LLM to generate answers, the loss is calculated, and the parameters of the low-rank matrix are updated by gradient descent.

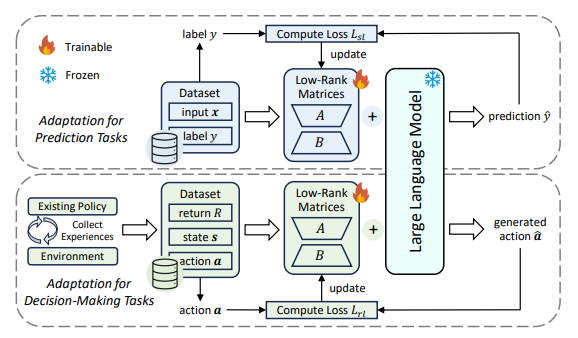


Figure 8: Illustration of the data-driven low-rank networking adaptation scheme of NetLLM.