RNA Toehold Switch Design via Diffusion Generation, Prediction and Biophysical Optimization

Overview

This repository implements a complete computational pipeline for **RNA toehold switch design**, integrating:

- Diffusion-based generation of RNA sequences
- Prediction models for ON / OFF translation strength
- Biophysical optimization guided by RNA secondary structure and thermodynamics

The code is organized to emphasize **methodological clarity and reproducibility**. This document is intended to help reviewers quickly understand the structure, purpose, and execution flow of the project.

Overall Pipeline

The overall workflow of the project is:

Diffusion-based generation
↓
Prediction of ON / OFF translation strength
↓
Biophysical optimization (structure and energy guided)

This order reflects the actual design logic and execution order of the codebase.

Generation Module: Diffusion Model

Purpose

The generation module learns a distribution over valid RNA toehold switch sequences using a **denoising diffusion probabilistic model (DDPM)**.

Main Entry

`train_cifar.py`

This script:

- Trains a UNet-based diffusion model
- Uses linear or cosine noise schedules
- Applies exponential moving average (EMA) for stable sampling
- Optionally aligns generated sequences with RNAErnie embeddings
- Periodically outputs generated RNA sequences

Prediction Module: Translation Strength

Purpose

The prediction module maps a full-length RNA toehold switch sequence (115 nt) to:

- ON translation strength
- OFF translation strength
- ON/OFF ratio

These predictions are used both for evaluation and as objective functions during optimization.

Main Training Scripts

- `on_structure_test1.py`: trains the ON translation prediction model
- `off_structure_test1.py`: trains the OFF translation prediction model

Prediction Interface

`switch_predict.py` provides a unified interface that:

- Accepts RNA sequences as input
- Outputs ON, OFF, and ON/OFF ratio
- Supports batch prediction and optional RNAErnie embeddings

This interface is reused by the optimization module and API service.

Biophysical Optimization Module

Purpose

This module refines candidate RNA sequences using black-box optimization guided by biological constraints.

Main Entry

`main_opt_biophysical.py`

Optimization is performed using **CMA-ES**, with objectives including:

- Predicted ON translation strength
- Predicted OFF translation strength
- ON/OFF ratio
- GC content constraints
- RNA secondary structure free energy (MFE)
- $\Delta\Delta G$ between alternative structural states

Key Utility Scripts

`make_synthetic_toehold_switch.py`

This script constructs synthetic RNA toehold switch sequences by assembling modular components (loops, stems, linker, start codon, etc.). It ensures fixed sequence length (115 nt) and consistent formatting across all modules.

`rna_switch_energy.py`

This module performs RNA secondary structure and thermodynamic analysis, including:

- RNAfold / RNAcofold calls
- Minimum free energy (MFE) computation
- $\Delta\Delta G$ calculation
- Structure parsing for optimization constraints

It is a core dependency of the biophysical optimization stage.

RNAErnie Large Model

This project optionally integrates **RNAErnie**, a pretrained RNA language model, to extract contextual RNA embeddings.

- Used for prediction models
- Used for embedding alignment during diffusion training

Official repository: https://github.com/CatIIIIIIII/RNAErnie

Embedding Cache

RNAErnie embeddings are generated on-the-fly during the first run and cached in the directory:

```
ernie_embeddings_batches/
```

This directory is **not provided beforehand** and is automatically created during execution to avoid repeated forward passes of the large pretrained model.

Environment Setup

System Requirements

- Linux (recommended)
- NVIDIA GPU with CUDA support
- Python $\geq$ 3.8

Step 1: Create Conda Environment

```
conda create -n rna_switch python=3.9
conda activate rna_switch
```

Step 2: Install Core Python Dependencies

```
pip install torch torchvision
pip install numpy pandas scipy tqdm
pip install matplotlib seaborn
pip install biopython
pip install cma
pip install fastapi uvicorn
```

Step 3: Install ViennaRNA (Required)

ViennaRNA is required for RNA secondary structure and free energy computation.

```
conda install -c bioconda viennarna
```

Verify installation:

```
python -c "import RNA; print(RNA.__version__)"
```

Step 4: Install RNAErnie Dependencies (Optional)

RNAErnie requires PaddlePaddle and PaddleNLP.

```
pip install paddlepaddle-gpu
pip install paddlenlp
```

The following files are included in RNAErnie's official repository:

- RNAErnie vocabulary file
- Pretrained RNAErnie checkpoint

Paths are configured inside the code.

Running the Pipeline

A typical execution order is:

1. Train diffusion model:

   ```
   python train_cifar.py
   ```

2. Train prediction models:

   ```
   python on_structure_test1.py
   python off_structure_test1.py
   ```

3. Run biophysical optimization:

   ```
   python main_opt_biophysical.py
   ```

Notes for Reviewers

- Datasets are not included due to size and licensing constraints
- All major methodological components are clearly separated
- Large-model embeddings are cached for efficiency and reproducibility
- The code emphasizes methodological transparency

License

This project is released under the MIT License.

```
MIT License

Copyright (c) 2024

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files...
```