Supplementary Materials

Dilith Jayakody
dilith.18@cse.mrt.ac.lk
Thanuja Ambegoda
thanuja@cse.mrt.ac.lk

Department of Computer Science and Engineering
University of Moratuwa
Moratuwa, Sri Lanka

A Implementation Details

A.1 Adaptations

Pretraining for valid output generation. Each iteration of MCTS spends a non-negligible amount of time evaluating the expression. Since the expression validity can be evaluated with a $(1,1,n_{channels})$-shaped tensor, the expression evaluation time can be significantly reduced during the pretraining stage. As a result, the model can learn to generate valid expressions much faster. This also gives the added benefit of providing a pretrained model to initialize weights for a new task, assuming the new task works with the same number of channels.

During the pretraining phase, certain tendencies are observed in the behavior of the agent. Firstly, the agent often generates short expressions. This may be because the more complex the expression, the easier it is to deviate from a regular range of values. Secondly, the agent tends to avoid opening parentheses. This is likely caused by the fact that once an opening parenthesis is generated, the expression remains invalid until a closing parenthesis is generated and this leads to a higher chance of negative rewards.

It is also observed that a significant fraction of existing remote-sensing indices is "unitless". In other words, if each channel of the multispectral image is given a unit of measurement, the spectral index resulting from a mathematical operation between those bands lacks a unit.

Accordingly, to motivate the agent to address the aforementioned considerations, the reward for pretraining $r(s)$ is defined as presented in equation 4.

$$r_{len} = 0.02 \times l_{exp} \tag{1}$$

$$r_{par} = 0.2 \times n_{par} \tag{2}$$

$$r_{unit} = \begin{cases} 1, \text{ if expression is unitless} \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

$$r(s) = 0.5 + r_{len} + r_{par} + r_{unit} \tag{4}$$

where l_{exp} is the length of the expression and n_{par} is the number of pairs of parentheses in the expression.

Previous Action	Valid Actions
$start, (, +, -, \times, /$	$(, channel$
$channel,)$	$+, -, \times, /,), =$

Table 1: The list of valid actions, given the previous action, where *start* refers to the starting state (empty expression) and *channel* refers to a channel of the image.

Action validity. At each state, we define a set of valid actions based on the previous action as shown in Table 1. In addition, certain other checks are performed to avoid invalidity and redundancy where possible.

- The number of ")" symbols in the expression is always maintained to be less than or equal to the number of "(" symbols in the expression. In other words, generating the ")" symbol is defined to be invalid if all opened parentheses have already been closed.
- Since enclosing a single symbol within parentheses is redundant as the parentheses can simply be removed, generating a closing parenthesis, two actions after generating an opening parenthesis, is defined to be an invalid action.

Adaptive Data Buffer. The classification of expressions as high-reward or low-reward can be achieved by various criteria. For the experiments performed in this research, this classification is performed by using an adaptive data buffer. The idea is to progressively reduce the size of the data buffer to get the GPT-based model to overfit to a set of expressions that provide a higher reward. We chose to implement this functionality as follows. The data collection phase is initially executed until the buffer reaches a certain capacity. The iterations that follow execute both the data collection phase and training phase. After each iteration, the buffer size is set to 95% of its current capacity, dropping the expressions within the 5% of lowest rewards. This is repeated until a certain minimum capacity is reached. This minimum capacity would be a relatively low number (e.g.: 20), such that the model may overfit to those expressions while also exploring expressions that contain similar symbols and structures.

A.2 Evaluated Index Preprocessing

Prior to being used in reward calculation, the evaluated index is updated by standardizing, clipping, and scaling to a [0, 1] range (Eq. 5 - 7).

$$standardize(\mathcal{E}) = \frac{\mathcal{E} - \mu_{\mathcal{E}}}{\sigma_{\mathcal{E}}} \tag{5}$$

$$clip(\mathcal{E}) = max\{min\{\mathcal{E}, Z_{max}\}, Z_{min}\} \tag{6}$$

$$scale(\mathcal{E}) = \frac{\mathcal{E} - Z_{min}}{Z_{max} - Z_{min}} \tag{7}$$

where $\mathcal{E}$ is the evaluated index, $\mu_{\mathcal{E}}$ and $\sigma_{\mathcal{E}}$ are the channel-wise means and standard deviations of $\mathcal{E}$, and Z_{min} and Z_{max} are the minimum and maximum Z-scores permitted. We use $Z_{min} = -3$ and $Z_{max} = 3$ in our experiments.

Function	Correlation	t-statistic	p-value
IoU	0.0554	0.7807	0.4359
CS	0.0599	0.8444	0.3995
F1	0.0667	0.9406	0.3481
AUC	0.0776	1.0952	0.2748
PCC	**0.1417**	**2.0142**	**0.0453**

Table 2: A statistical comparison of the correlations of each heuristic function with the training score

Size	Baseline	NDVI		Generated	
		R	RM	R	RM
UNet	58.0	74.0	73.6	73.6	73.7
UNet++	60.9	70.2	72.4	73.6	70.3

Table 3: **Effects of Generated Indices versus Existing Indices**. The table shows the comparison between the IoU scores of the multiple-index replacement method (RM) and the single-index replacement method (R) for the NDVI index and the index generated by the proposed method for the Grass class of the RIT-18 dataset.

A.3 GPT-based Model Configurations

A.3.1 Model Architecture

Number of layers = 4
Number of attention heads = 4
Embedding size = 128
Dropout = 0.0

A.3.2 Adam Optimizer

Learning rate = $1e-4$
Weight decay = 0.1
Beta1 = 0.9
Beta2 = 0.95

B Additional Experiments

B.1 Qualitative Comparison

We perform a qualitative comparison between the baseline (B) model and the multi-index replacement (RM) mode (Figure 1). Through the comparison, we observe that the RM mode tends to be relatively less confused by objects that blend into the background in terms of color.

B.2 Effects of Generated Indices versus Existing Indices.

To evaluate the performance of the generated indices in comparison to pre-existing indices, we compare NDVI to the expression for the Grass class of the RIT-18 dataset. We choose this setting due to the specific utility of NDVI in the identification of greenery in prior work.

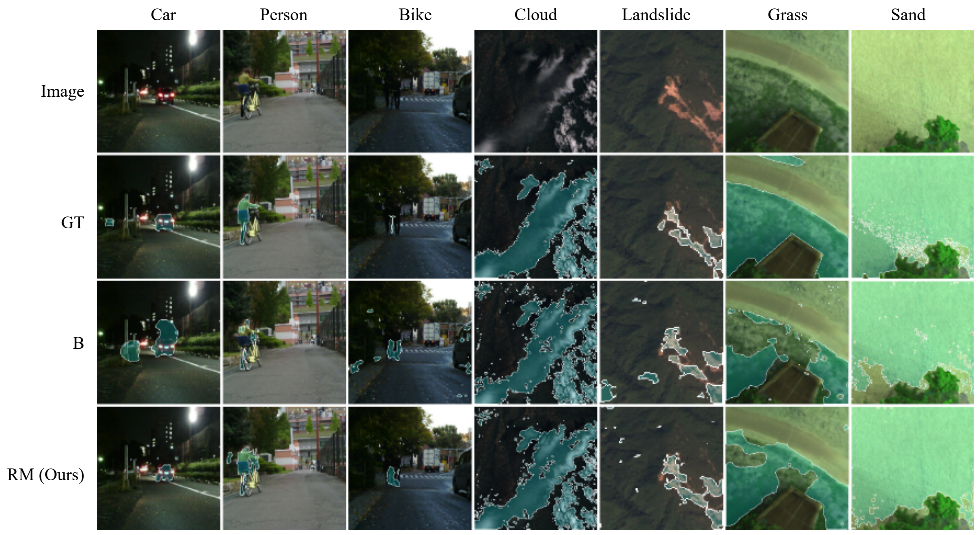

Figure 1: Qualitative Comparison between the baseline result (B) and Multi-index replacement (RM) across the datasets with the UNet model, along with the ground truth (GT).

Despite the NDVI index being specifically created for this context, we observer comparable results when using the generated indices

C Code

Please note that the codebase for the project is available in the supplementary material in the `code.zip` file. The `README.md` file within the codebase provides instructions on how to set up the workspace, download the dataset, and execute the algorithms.

Once the dataset is downloaded and extracted, the resulting directory structure is expected to look as follows:

```
./
|- dataset/
|- indexrl/
|_ dataset.py
|_ dataset_config.py
|_ ...
```