Lumina AI

Getting Started with PrismRCL – 2.7.2

PrismRCL is a foundational AI algorithm application for Windows and Linux/Darwin. Below are basic prepping, training, and inference examples to get you started with PrismRCL. Before you start training PrismRCL on your data, please make sure that it is organized and shaped correctly. Our algorithm supports three types of data: images in the PNG format, text data, and tabular data. Each text or tabular sample must be in its own file. For tabular data, please make sure that for each sample, the features are space-separated and stored in a single line text file. Starting with version 2.4.0, tabular data values no longer need to be normalized.

Note: Please review the "readtextbyline" parameter in section 3 below for special LLM dataset formatting requirements.

Required structure of your dataset (non-LLM):

```
parent_folder/
            class1_folder/img1.png, img2.png, etc.
            class2_folder/
            ...

parent_folder/
            class1_folder/file1.txt, file2.txt, etc.
            class2_folder/
            ...
```

Note: file names across all class folders must be unique!

Important: for LLM datasets, each class folder must contain a single text file that includes all samples belonging to that class; one sample per line.

1. Installation and Activation:

PrismRCL does not require installation. Just unzip the files included in the package into a single folder on your hard drive and you're ready to go! Make sure that the location of your executable is added to the system PATH, or you may need to change directories to where the binary is before the application will run correctly. Please note that PrismRCL is intended as a developer tool that you can build applications and scripts on top of. It is designed to be run from the command line as shown in the examples below. It does not have a graphical user interface. All logs and outputs are directed to text files on the file system and are fully configurable by the user. This makes it easy to run PrismRCL in batch mode and allows for automation and the ability to run machine learning tasks unattended.

PrismRCL comes with a free 30-day trial period. If you've purchased a subscription from us, please use your license key anytime during the trial to activate the software. It is recommended that you run the application once without any parameters to initialize the trial period (Windows or Linux/Darwin).

New in version 2.7.1: PrismRCL now supports MacOS (Darwin) and ARM processors. It has been successfully tested on Windows (x86_64 and aarch64), Linux (x86_64 and aarch64), and Darwin (x86_64 and aarch64).

New in version 2.7.0: PrismRCL now supports Linux. It has been successfully tested on Ubuntu 22 & 24, as well as Red Hat Enterprise 9 and 10 and Fedora Workstation 42.

Note: You may encounter the following error on Ubuntu 22 (especially in WSL) when you run prismrcl on your system for the first time: **GTK 3 is required to be installed.** If you do, please follow the following steps to resolve the issue:

UBUNTU:

sudo apt update

sudo apt install libgtk-3-dev mesa-utils libgl1-mesa-glx ttf-mscorefonts-installer

RED HAT:

sudo yum install gtk3 gtk3-devel mesa-dri-drivers

This will install GTK 3 and related dependencies required for prismrcl.

New in version 2.5.3: PrismRCL can now be run in a Docker container. It was tested on Windows. The containerized version is available only for Enterprise clients.

Note: the examples in this document show how to build your commands in Windows, but the Linux/Darwin commands are built in a similar way. The only difference is that the paths are specified in the Linux/Darwin style. Here is an example to illustrate:

Windows: C:\PrismRCL\PrismRCL.exe auto-optimize data=C:\data\train_data log=c:\log_files\

Linux/Darwin: /home/prismrcl/prismrcl auto-optimize data=/home/data/train_data log=/home/log_files/

2. Finding the optimal training parameters for your dataset (Windows example):

Before you start training the algorithm on your data, you will need to set some parameters that will optimize the training and result in the best model for your data. PrismRCL offers a special feature that can find the optimal training parameters for your dataset automatically. We call it **auto-optimize**. No more trying random parameters endlessly. Let PrismRCL do the work for you! The three examples below demonstrate how you can use this time saving and optimization tool.

New in version 2.5.0: auto-optimize will automatically pursue different goals as follows:

Two classes: Overall Accuracy (acc)

If the dataset is reasonably balanced: Macro Average F1 Score (af1)

If the dataset is imbalanced: Weighted Average F1 Score (wf1)

Note: in addition to the functionality above, auto-optimize can also be forced to pursue the desired goal, by passing the auto-optimize parameter in this format:

auto-optimize=acc (auto-optimize will pursue Overall Accuracy)

auto-optimize=af1 (auto-optimize will pursue Macro Average F1 Score)

auto-optimize=wf1 (auto-optimize will pursue Weighted Average F1 Score)

auto-optimize=mcc (auto-optimize will pursue Matthews Correlation Coefficient)

The command below will find and save the parameters to the log directory in a file of this format: **_optimize_summary_mm_dd_yy_hh_mm_ss.txt**.

C:\PrismRCL\PrismRCL.exe auto-optimize data=C:\data\train_data log=c:\log_files

data: path to the training image/text dataset shaped as explained above (non-LLM).

log: path to where PrismRCL will save the training session log and results files.

The command below will find and save the parameters to the log directory in a file of this format: **_optimize_summary_mm_dd_yy_hh_mm_ss.txt**.

C:\PrismRCL\PrismRCL.exe auto-optimize=mcc data=C:\data\train_data log=c:\log_files

auto-optimize=mcc: auto-optimize will pursue the maximization of mcc (Matthews Correlation Coefficient). This is a new feature in version 2.5.0. If not optimization goal is specified, PrismRCL will automatically determine the goal based on the data distribution.

data: path to the training image/text dataset shaped as explained above (non-LLM).

log: path to where PrismRCL will save the training session log and results files.

The command below will create a model using parameters from auto-optimize (standard optimization parameters *and possibly imaginaryslice* are stored in the model).

C:\PrismRCL\PrismRCL.exe auto-optimize data=C:\data\train_data savemodel=c:\models\best_model.model log=c:\log_files

data: path to the training image/text dataset shaped as explained above (non-LLM).

savemodel: path to where PrismRCL will save the trained model.

log: path to where PrismRCL will save the training session log and results files.

The command below will create a model using parameters from auto-optimize (evaluation method, rclticks, boxdown, *and possibly imaginaryslice* are stored in model) and evaluates model on test data, all in one pass.

C:\PrismRCL\PrismRCL.exe auto-optimize data=C:\data\train_data testdata=C:\data\test_data savemodel=c:\models\best_model.model log=c:\log_files

data: path to the training image/text dataset shaped as explained above (non-LLM).

testdata: path to the test dataset to be used for evaluating the model during training.

savemodel: path to where PrismRCL will save the trained model.

log: path to where PrismRCL will save the training session log and results files.

IMPORTANT: For LLM datasets, your auto-optimize command should look like this:

C:\PrismRCL\PrismRCL.exe llm readtextbyline auto-optimize data=C:\data\train_data log=c:\log_files

Note: the "readtextbyline" requirement for LLM formatted datasets is explained in section 3 below.

3. Create a new LLM training session (Windows example):

New in version 2.6.0: PrismRCL now supports training text datasets for language modeling. We introduce a new parameter called "llm" which will instruct PrismRCL to treat the training dataset as a Large Language Model dataset.

The command below will start an LLM training session given a training dataset and a test split size.

C:\PrismRCL\PrismRCL.exe llm naivebayes directional rclticks=67 readtextbyline data=C:\PrismRCL\data\llm-dataset\train-data testsize=0.1 savemodel=C:\PrismRCL\models\llm-model.model log=C:\PrismRCL\logfiles\job_folder stopwhendone

The command below will start an LLM training session given a training dataset and a test dataset.

C:\PrismRCL\PrismRCL.exe llm naivebayes directional rclticks=67 readtextbyline data=C:\PrismRCL\data\llm-dataset\train-data testdata= C:\PrismRCL\data\llm-dataset\test-data savemodel=C:\PrismRCL\models\llm-model.model log=C:\PrismRCL\logfiles\job_folder stopwhendone

The evaluation method for both training sessions is **naivebayes** and **rclticks** is set.

llm: instructs PrismRCL to treat the text training data as an LLM dataset.

directional: instructs PrismRCL to maintain the order of the features in a data sample. In this case, to maintain the order of words in an input text sequence.

readtextbyline: instructs PrismRCL to treat each text file it loads data from as containing multiple samples, one sample per line. This is the standard format required for LLM training datasets.

data: path to the training image dataset shaped as explained above.

testsize: fraction of training dataset to be used for testing the model during training.

testdata: path to the test dataset to be used for evaluating the model during training.

savemodel: path to where PrismRCL will save the trained model.

log: path to where PrismRCL will save the training session log and results files.

stopwhendone: instructs PrismRCL to shutdown automatically once the training session ends.

Tip: when building your command to create a new model, we suggest that you give your model a descriptive name to make it easier to identify later. For example, if you are training your model on breast cancer biopsy images using version 2.7.1 of the software, you might want to call your model something like:

rcl-llm_v271_09052025.model

Note: Only the "naivebayes" and "softmax" evaluation methods are currently supported for LLM training. Also, "directional" should always be specified since the order of words in language processing always matters.

4. Create a new training session with test split (Windows example):

The command below will start a training session given a training dataset and a test split size.

C:\PrismRCL\PrismRCL.exe fractal rclticks=15 data=C:\PrismRCL\data\dataset\train-data testsize=0.1 savemodel=C:\PrismRCL\models\model_name.model log=C:\PrismRCL\logfiles\job_folder stopwhendone

The evaluation method for this training session is **fractal** and **rclticks** is set (used for image data only).

data: path to the training image dataset shaped as explained above.

testsize: fraction of training dataset to be used for testing the model during training.

savemodel: path to where PrismRCL will save the trained model.

log: path to where PrismRCL will save the training session log and results files.

stopwhendone: instructs PrismRCL to shutdown automatically once the training session ends.

Tip: when building your command to create a new model, we suggest that you give your model a descriptive name to make it easier to identify later. For example, if you are training your model on breast cancer biopsy images using version 2.4.2 of the software, you might want to call your model something like:

breast-cancer-biopsy_v242_03112024.model

Notes:

Models created with version 2.7.x are not backwards compatible with previous versions of PrismRCL.

Models created with earlier versions (prior to 2.7.x) are not compatible with version 2.7.x of PrismRCL.

5. Create a new training session with test dataset (Windows example):

The command below will start training session given a training dataset and a test dataset.

C:\PrismRCL\PrismRCL.exe chisquared rclticks=15 data=C:\PrismRCL\data\dataset\train-data testdata= C:\PrismRCL\data\dataset\test-data savemodel=C:\PrismRCL\models\model_name.model log=C:\PrismRCL\logfiles\job_folder stopwhendone

This is very similar to the training command above. The only difference is that it uses a separate test dataset for evaluating the model, as opposed to using a split out of the training data. The evaluation method for this training session is **chisquared** and **rclticks** is set.

data: path to the training image dataset shaped as explained above.

testdata: path to the test dataset to be used for evaluating the model during training.

savemodel: path to where PrismRCL will save the trained model.

log: path to where PrismRCL will save the training session log and results files.

stopwhendone: instructs PrismRCL to shutdown automatically once the training session ends.

6. Create a new transfer learning session with test split (Windows example):

The command below will start transfer learning training session given a training dataset and a test split size.

C:\PrismRCL\PrismRCL.exe naivebayes rclticks=15 transferlearn=c:\PrismRCL\models\existing_model_name.model data=C:\PrismRCL\data\dataset\train-data testsize=0.1 savemodel=C:\PrismRCL\models\new_model_name.model log=C:\PrismRCL\logfiles\job_folder stopwhendone

In this example, PrismRCL will use a pretrained model to train further on more training data. The evaluation method for this training session is **naivebayes** and **rclticks** is set.

transferlearn: path to the pretrained model to be used as the starting point for transfer learning session.

data: path to the training image dataset shaped as explained above.

testsize: fraction of training dataset to be used for testing the model during training.

savemodel: path to where PrismRCL will save the trained model.

log: path to where PrismRCL will save the training session log and results files.

stopwhendone: instructs PrismRCL to shutdown automatically once the training session ends.

7. Combine two or more models into one (Windows example):

The command below will combine three models together and save them into one final model. Please note that when adding models together, make sure that the individual models were created using chunks from the same dataset and using the same training parameters. That's what makes it possible for the models to be combined.

C:\PrismRCL\PrismRCL.exe loadmodel=c:\PrismRCL\models\bettermodel.model addmodel=c:\PrismRCL\models\model998.model;c:\PrismRCL\models\model_999.model savemodel=c:\PrismRCL\models\bestmodel.model stopwhendone

PrismRCL can combine or "add up" models that were created in separate training sessions. To add models, all you need is to specify the locations on disk where your base model and additional models are. No parameters are needed.

loadmodel: path to the pre-trained model to be used as the starting point for transfer learning session.

addmodel: paths to the additional models to be combined with the based training image dataset shaped as explained above. Model paths must be separated by semicolons.

savemodel: path to where PrismRCL will save the trained model.

stopwhendone: instructs PrismRCL to shutdown automatically once the training session ends.

8. Create a new inference session (Windows example):

The command below will start an inference session using a pre-trained model and an inference dataset (not used in training or testing the model).

C:\PrismRCL\PrismRCL.exe loadmodel= C:\PrismRCL\models\best_model.model testdata=D:\deploy\data\dataset\inference-data inftotext=C:\PrismRCL\output\inference_output.txt log=D:\PrismRCL\logfiles\job_folder stopwhendone

New in version 2.5.0: When running inference, you no longer need to specify the training parameters. PrismRCL now saves those parameters in the model during training, and they are loaded automatically when creating an inference session.

Note: If running inference on an LLM model, add "**llm**" and "**readtextbyline**" to your command (*ensure that your test dataset is shaped correctly. See section 3 above for more information*).

loadmodel: path to the pre-trained model to be used as the starting point for transfer learning session.

data: path to the training image dataset shaped as explained above.

testdata: path to the inference dataset that you'd like your pre-trained model to classify.

inftotext: path to the output text file where predicted classes for all inference images will be saved. The output file will contain two columns: name of image and predicted class.

log: path to where PrismRCL will save the inference session log and results files.

stopwhendone: instructs PrismRCL to shutdown automatically once the training session ends.

9. Create a new inference web instance (Windows example):

Note: PrismRCLM now supports SSL (starting with version 2.7.2). For complete instructions on how to start PrismRCLM in SSL mode, please refer to the included technical documentation.

Using PrismRCLM.exe, the command below will load a model into a web-based inference instance. The result is a memory-resident model that can be used for fast inference using HTTP requests.

C:\PrismRCL\PrismRCLM.exe loadmodel=c:\models\model111.model port=8080

In the example above, the instance will be accessible via **http://localhost:8080/**

Tip: Multiple models can be loaded on the same physical (or virtual) server using different port numbers. A model will be identified by the port number assigned to the web service when it was started.

The PrismRCLM web instance will respond with one of the following four statuses:

ready: if called with no parameters, and if server is idle.

done: when server has finished processing a request.

busy: if server is busy processing another request.

error: if request contains an error and server could not process it.

To run inference using PrismRCLM, a web request is made on the running instance of choice. The port number will indicate the model to be used for inference. In addition, the evaluation type, the input data, and the

prediction output must be specified. You are encouraged to generate log files for your inference sessions, although this is not required. The following are some examples.

Inference to text on an image data model:

http://locahost:8080/evaluation_method&imaginary_flag&testdata=C:/RCLC/data/test_data/&log=C:/RCLC/logfiles/&inftotext=C:/RCLC/output/prediction_output.txt

IMPORTANT: Paths specified in the request URL must be Windows paths and must be accessible locally on the system where the PrismRCLM instance is running (network drives are supported).

port number: indicates the port number that your web instance will listen on. Multiple models can be identified by their port numbers.

evaluation_method: chisquared | fractal | naivebayes | softmax

Note: Evaluation method must match the method used for training the model.

imaginary_flag: &imaginary if enabled, blank otherwise.

testdata: path to folder containing samples to run inference on.

log: path to folder where log files for the inference session will be stored.

inftotext: path to file where predictions will be stored. The file will contain a list of inference samples and their predicted classes.

Inference to image on an image data model:

http://locahost:8080/evaluation_method&imaginary_flag&testdata=C:/RCLC/data/test_data/&log=C:/RCLC/logfiles/&inftoimage=C:/RCLC/output/predictions/

inftoimage: path to folder where augmented image predictions are stored (see parameter explanation in section 2 for more details.)

Inference to text on a text or tabular data model:

http://locahost:8080/evaluation_method&imaginary_flag&testdata=C:/RCLC/data/test_data/&log=C:/RCLC/logfiles/&textinftotext=C:/RCLC/output/prediction_output.txt

textinftotext: path to file where predictions will be stored. The file will contain a list of text or tabular inference samples and their predicted classes.

Inference to text on a text LLM data model:

http://locahost:8080/evaluation_method&imaginary_flag&llm&readtextbyline&testdata=C:/RCLC/data/test_data/&log=C:/RCLC/logfiles/&textinftotext=C:/RCLC/output/prediction_output.txt

evaluation_method: naivebayes | softmax