ONNX Interpreter

Perlo Giacomo 317981
Sanino Fabrizio 317541

academic year 2022/23

Summary

Overview

Onnx Parser

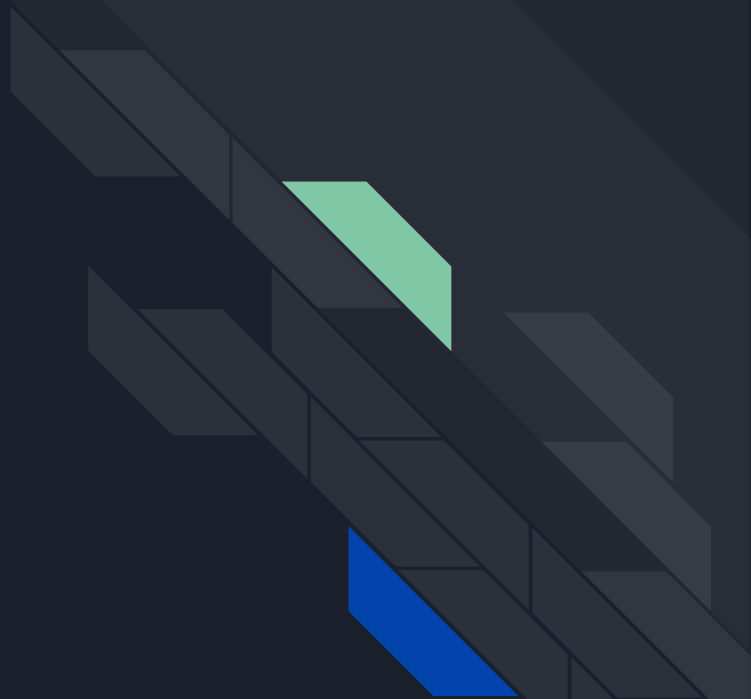
Onnx Serializer

Inference

Multi-Threading

Binding

Results



Overview

ONNX is an open format built to represent machine learning models.

ONNX defines a common set of operators and a common file format to enable AI developers to use models with a variety of frameworks, tools, runtimes, and compilers.

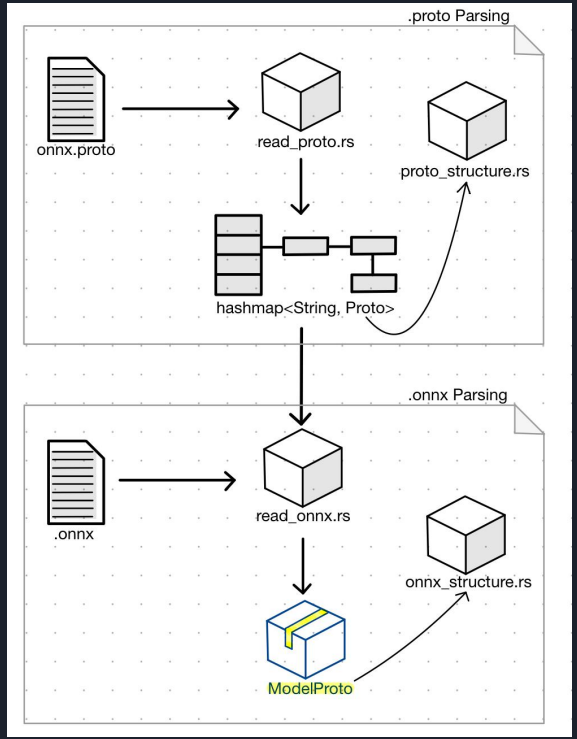
The objective of the project is to build a ONNX Interpreter using the **Rust** language.
The project, indeed, provides:

- a ONNX Model Parser and Serializer
- the opportunity to make Inference on the parsed model by leveraging Multi-Threading paradigm
- the binding towards Python of the Inference functionality

Onnx Parser

The objective of the Parser is to obtain the runtime Onnx model starting from its .onnx file.
The parser work could be split in two main phases:

- ***.proto*** parsing, the ***onnx.proto*** file is a *protocol buffer* file which defines the structure of .onnx files and, moreover, allows them to be read. Its content is read and then stored in a hashmap.
- ***.onnx*** parsing, this phase allows the parser to generate the runtime Onnx model starting from the ***.onnx*** file together with the hashmap defined in the previous phase.



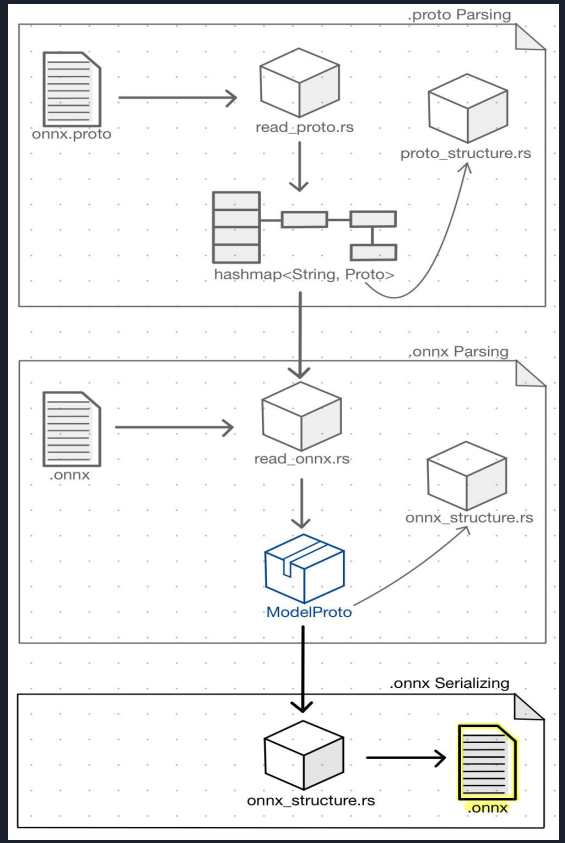
Onnx Serializer

The objective of the Serializer is to write a new .onnx file starting from a runtime onnx model previously read.

Furthermore, this module allows to modify the model such as creating new nodes or changing other features.

The goal is reached by exploiting the ***onnx_structure.rs*** file.
This one was automatically generated by ***rust-protobuf 3.2.0*** together with the ***onnx.proto***.

Therefore the ***onnx_structure.rs*** file, once consistently adapted to our specific needs, allowed to write back on a .onnx file the runtime model, scanning its content and serializing it into a binary format.



Inference

The objective of the **inference** is to execute the *.onnx* model's operation on the input data.

Based on the ***op_type*** of the node, we can distinguish which operation must be performed.

The most significant **operations** we have developed (based on the models that we decided to execute) are:

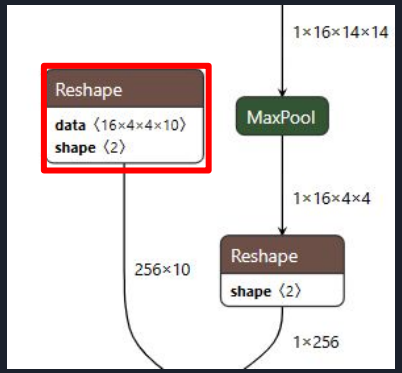
- Convolution: for a given input (2d image) it allows to obtain different convolved features based on the filter applied on the input
- MaxPool: for a given input (2d image) it allows to calculate the maximum value for patches of the image and uses it to create a downsampled map
- Relu: Activation function

Other operations developed: Dropout, GlobalAvergaPool, Reshape, Softmax.

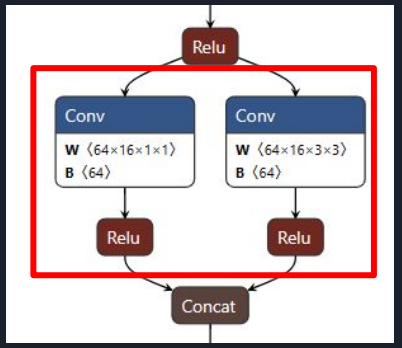
Multi Threading

The objective of exploiting **Multi Threading** is to speed up the Inference process. The two scenarios in which this is possible are:

- **Independent nodes** nodes whose data is available from the beginning of the inference process (since stored in the Model *Initializers).*



- **Parallel nodes** nodes who share the same input data.



Every time a node (or a group of nodes) respect one of the aforementioned characteristics, a thread (or a group of threads) is spawned. Whenever the following process doesn't have the data needed to execute the inference, then it is suspended on a **condition variable**.

Binding

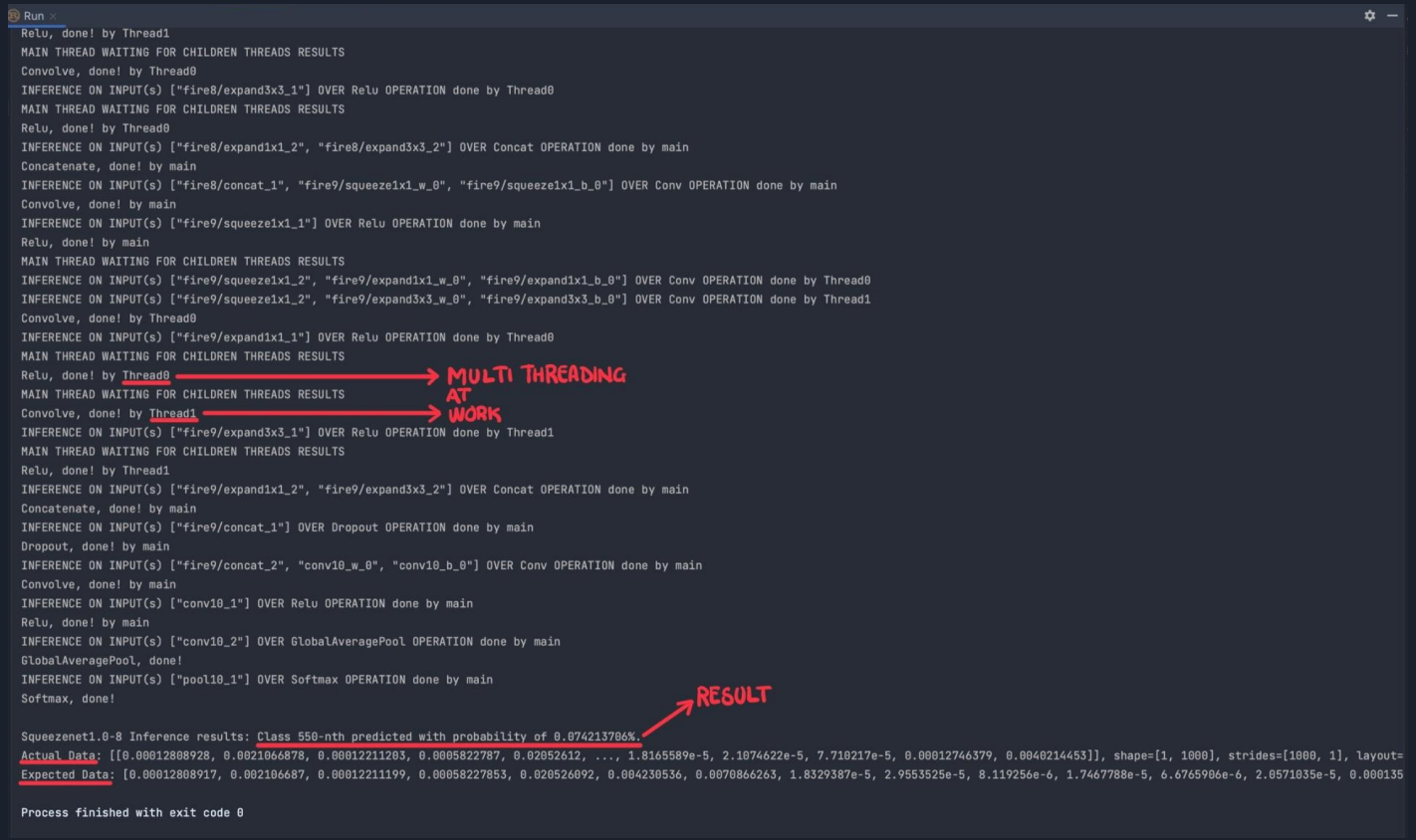
The objective of the Binding module is to give the possibility of using and exporting functions written in a language into another programming language. It was decided to export Rust functions to **Python**.

The technologies used are the following:

- **py03**: is a Rust bindings for the Python interpreter. After importing it in our project, functionalities that we had want to be exported were wrapped with a specific methods present in those library.
- **maturin**: tool that is used for build and publish Python package.
- **venv**: tool that allows to create Python virtual environment.

For simplicity we had chosen to use *maturin develop* command which allows to build the crate and installs it as a python module directly in the **virtulenv**.

Results


Run
Relu, done! by Thread1
MAIN THREAD WAITING FOR CHILDREN THREADS RESULTS
Convolve, done! by Thread0
INFERENCE ON INPUT(s) ["fire8/expand3x3_1"] OVER Relu OPERATION done by Thread0
MAIN THREAD WAITING FOR CHILDREN THREADS RESULTS
Relu, done! by Thread0
INFERENCE ON INPUT(s) ["fire8/expand1x1_2", "fire8/expand3x3_2"] OVER Concat OPERATION done by main
Concatenate, done! by main
INFERENCE ON INPUT(s) ["fire8/concat_1", "fire9/squeeze1x1_w_0", "fire9/squeeze1x1_b_0"] OVER Conv OPERATION done by main
Convolve, done! by main
INFERENCE ON INPUT(s) ["fire9/squeeze1x1_1"] OVER Relu OPERATION done by main
Relu, done! by main
MAIN THREAD WAITING FOR CHILDREN THREADS RESULTS
INFERENCE ON INPUT(s) ["fire9/squeeze1x1_2", "fire9/expand1x1_w_0", "fire9/expand1x1_b_0"] OVER Conv OPERATION done by Thread0
INFERENCE ON INPUT(s) ["fire9/squeeze1x1_2", "fire9/expand3x3_w_0", "fire9/expand3x3_b_0"] OVER Conv OPERATION done by Thread1
Convolve, done! by Thread0
INFERENCE ON INPUT(s) ["fire9/expand1x1_1"] OVER Relu OPERATION done by Thread0
MAIN THREAD WAITING FOR CHILDREN THREADS RESULTS
Relu, done! by Thread0
MULTI THREADING AT WORK
MAIN THREAD WAITING FOR CHILDREN THREADS RESULTS
Convolve, done! by Thread1
INFERENCE ON INPUT(s) ["fire9/expand3x3_1"] OVER Relu OPERATION done by Thread1
MAIN THREAD WAITING FOR CHILDREN THREADS RESULTS
Relu, done! by Thread1
INFERENCE ON INPUT(s) ["fire9/expand1x1_2", "fire9/expand3x3_2"] OVER Concat OPERATION done by main
Concatenate, done! by main
INFERENCE ON INPUT(s) ["fire9/concat_1"] OVER Dropout OPERATION done by main
Dropout, done! by main
INFERENCE ON INPUT(s) ["fire9/concat_2", "conv10_w_0", "conv10_b_0"] OVER Conv OPERATION done by main
Convolve, done! by main
INFERENCE ON INPUT(s) ["conv10_1"] OVER Relu OPERATION done by main
Relu, done! by main
INFERENCE ON INPUT(s) ["conv10_2"] OVER GlobalAveragePool OPERATION done by main
GlobalAveragePool, done!
INFERENCE ON INPUT(s) ["pool10_1"] OVER Softmax OPERATION done by main
Softmax, done!
RESULT
Squeezenet1.0-8 Inference results: Class 550-nth predicted with probability of 0.074213706%.
Actual Data: [[0.00012808928, 0.0021066878, 0.00012211203, 0.0005822787, 0.02052612, ..., 1.8165589e-5, 2.1074622e-5, 7.710217e-5, 0.00012746379, 0.0040214453]], shape=[1, 1000], strides=[1000, 1], layout=
Expected Data: [0.00012808917, 0.002106687, 0.00012211199, 0.00058227853, 0.020526092, 0.004230536, 0.0070866263, 1.8329387e-5, 2.9553525e-5, 8.119256e-6, 1.7467788e-5, 6.6765906e-6, 2.0571035e-5, 0.000135
Process finished with exit code 0