

ML Frameworks in Athena (onnxruntime)

Debottam BakshiGupta

UTA

12/04/2020

Index

- Use cases of ML frameworks in athena
- Building strategy of ML framework as a part of external software
- Resources available in athena to support existing industrial ML framework
- Best choice for atlas
- Current implementation of the framework
- Future plans

Use cases in athena

- We will be using trained neural network (NN) models in athena for prediction
 - No plans for training models in athena
- The ML framework to be used must have C++ API compatible with athena
- Possible thread safe application or thread exclusive processing
 - E.g. Tensorflow spawns lots of threads by default keeping CPU busy for itself only.
- Memory management during run
 - The NN models need to be contained in minimum memory otherwise will be incompatible with trigger related application.
- Last but not the least we don't want user to deal with complexities of core software
 - There should be a way to create an abstraction layer to communicate with the user

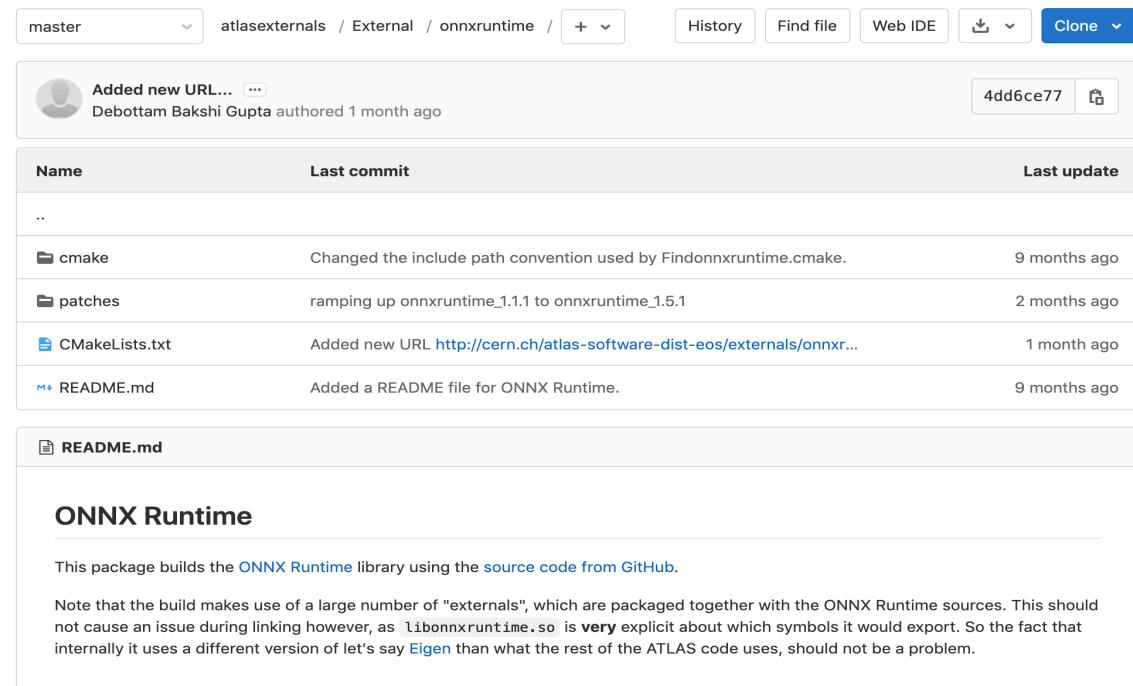
Building strategy and Favorites

Building Strategy: After checking its compatibility with athena build the ML framework as [ATLASexternals](#)

[Onnxruntime](#) is a great tool for parsing trained model. You can train your model in any platform (e.g. tensorflow, Keras, pytorch, etc.) and convert them to .onnx format using *platform-to-onnx (* tensorflow, Keras, pytorch, etc) API. Once your model is in .onnx format you can parse and predict with your model using onnxruntime.

Athena Compatibility:

- ✓ C++ API,
- ✓ Supports complex models (e.g. CNN, RNN, GAN, VAE),
- ✓ Thread exclusive processing



The screenshot shows the GitHub repository page for 'atlasexternals / External / onnxruntime'. The repository is on the 'master' branch. A commit by Debottam Bakshi Gupta, titled 'Added new URL...', is highlighted. Below the commit list, the README.md file is displayed, containing the following text:

```
ONNX Runtime

This package builds the ONNX Runtime library using the source code from GitHub.

Note that the build makes use of a large number of "externals", which are packaged together with the ONNX Runtime sources. This should not cause an issue during linking however, as libonnxruntime.so is very explicit about which symbols it would export. So the fact that internally it uses a different version of let's say Eigen than what the rest of the ATLAS code uses, should not be a problem.
```

Onnxruntime in athena

- Where in atlas:
<https://gitlab.cern.ch/atlas/atlasexternals/-/tree/master/External/onnxruntime>
- Version:
Onnxruntime_v1.5.1
- How to use in athena:
In packages' CmakeLists.txt
 - atlas_add_library
 - INCLUDE_DIRS \${ONNXRUNTIME_INCLUDE_DIRS}
 - LINK_LIBRARIES \${ONNXRUNTIME_LIBRARIES}
 - atlas_add_component
 - INCLUDE_DIRS \${ONNXRUNTIME_INCLUDE_DIRS}
 - LINK_LIBRARIES \${ONNXRUNTIME_LIBRARIES}

Running a session in athena: use the shared service located at [AthOnnxruntimeService](#). Example [package](#)

```
65     /// Handle to @c AthONNX::IONNXRuntimeSvc
66     ServiceHandle< IONNXRuntimeSvc > m_svc{ this, "ONNXRuntimeSvc",
67                                             "AthONNX::IONNXRuntimeSvc",
68                                             "Name of the service to use" };
```

Out of box build

- The usual cmake build of onnxruntime has a strong python 3 dependency
- The produced nightly builds may not have particular python 3 version
- A way around implemented by applying a [patch file](#)

atlas > atlasexternals > Repository

master atlasexternals / External / onnxruntime / patches History Find file Web IDE Clone

ramping up onnxruntime_1.1.1 to onnxruntime_1.5.1
Debottam Bakshi Gupta authored 2 months ago 85f34746

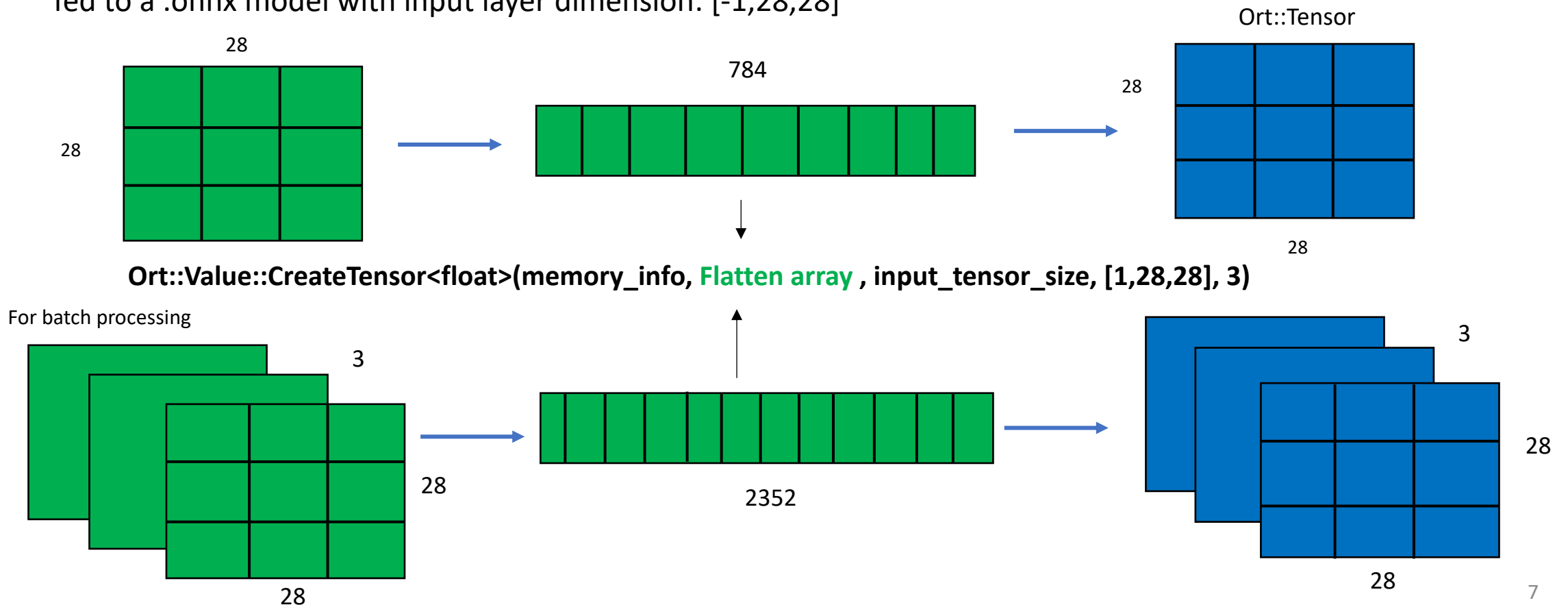
Name	Last commit	Last update
..		
onnxruntime-1.1.1-automl_featurizers.pat...	Turned off the unit tests during the onnxruntime build.	7 months ago
onnxruntime-1.1.1-cmake.patch	upgrading to current onnxruntime release v1.1.1 ("http...	9 months ago
onnxruntime-1.5.1-cmake.patch	ramping up onnxruntime_1.1.1 to onnxruntime_1.5.1	2 months ago

- Whenever switching to a new version of onnxruntime the patch files should be updated

Onnxruntime Tensor (Ort::Value)

.onnx model takes input in Ort::Value format; ([here](#))

e.g. MNIST hand written digit 3D array (1, 28, 28) needs to be converted to 3D Ort::Value before being fed to a .onnx model with input layer dimension: [-1,28,28]



Successful attempts in athena ([example](#))

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 512)	401920
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 10)	5130

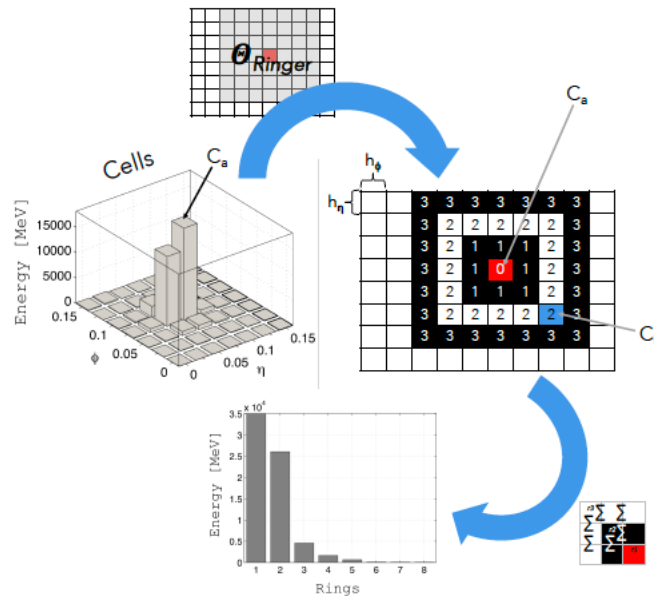
```
AthONNX          DEBUG Input 0 : name= flatten_input:0
AthONNX          DEBUG Input 0 : num_dims= 3
AthONNX          DEBUG Input 0 : dim 0= -1
AthONNX          DEBUG Input 0 : dim 1= 28
AthONNX          DEBUG Input 0 : dim 2= 28
AthONNX          DEBUG Output 0 : name= dense_1/Softmax:0
AthONNX          DEBUG Output 0 : num_dims= 2
AthONNX          DEBUG Output 0 : dim 0= -1
AthONNX          DEBUG Output 0 : dim 1= 10

INFO Label for the input test data = 1
AthONNX          DEBUG Score for class 0 = 1.07293e-07
AthONNX          DEBUG Score for class 1 = 0.999818
AthONNX          DEBUG Score for class 2 = 1.18024e-05
AthONNX          DEBUG Score for class 3 = 2.53529e-05
AthONNX          DEBUG Score for class 4 = 4.19157e-06
AthONNX          DEBUG Score for class 5 = 1.66088e-06
AthONNX          DEBUG Score for class 6 = 7.7723e-06
AthONNX          DEBUG Score for class 7 = 6.33801e-05
AthONNX          DEBUG Score for class 8 = 5.83467e-05
AthONNX          DEBUG Score for class 9 = 9.74693e-06
AthONNX          INFO Class: 1 has the highest score: 0.999818
```

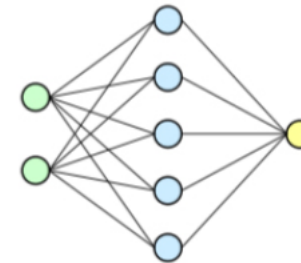

Physics Example (Ringer in Trigger electron)

Process in brief:

- Level 1: Provides RoI (θ_{RoI}) in $\eta \times \phi$ axis
- HLT : within the $\eta \times \phi$: 0.4×0.4 window of θ_{RoI} all rings are formed.
 - Ring_variables are summation of ET of all cells belonging to a ring around θ_{RoI}



100 such ET summations belonging layers $l \in \{PS, EM1, EM2, EM3, HAD1, HAD2, HAD3\}$ fed to a single layer DNN



Depending on the type of selection (loose, medium, tight) and η position there are several such trained model.

There is a scope for using CNN instead of DNN, currently being developed

WIP: RingerTool ([!35777](https://github.com/135777))

Parsing model through onnxruntime

```
RingerSelectorTool 5 0 INFO Input 0 : name= dense_166_input
RingerSelectorTool 5 0 INFO Input 0 : num_dims= 2
RingerSelectorTool 5 0 INFO Input0 : dim 0= 1
RingerSelectorTool 5 0 INFO Input0 : dim 1= 100
RingerSelectorTool 5 0 INFO Output 0 : name= dense_167
RingerSelectorTool 5 0 INFO Output 0 : type= 1
RingerSelectorTool 5 0 INFO Output 0 : num_dims= 2
RingerSelectorTool 5 0 INFO Output0 : dim 0= 1
RingerSelectorTool 5 0 INFO Output0 : dim 1= 1
RingerSelectorTool 5 0 INFO The current model predict with output: 2.22336
RingerSelectorTool 5 0 INFO Event et = 0.0200504, eta = 2.2236
RingerSelectorTool 5 0 INFO Output = 2.22336 Avgmu = 57.169
RingerSelectorTool 5 0 INFO Passed ? 1
```

Annotations:

- Input layer name
- 2D array
- Input Row size: 1
- Input Column size: 100
- Output layer name
- 2D array
- Output Row size: 1
- Output Column size: 1

Other projects being supported by onnxruntime

- Offline electron ID (details in Kazuya's [talk](#)) with following input (not in master yet)

```
images    = ['em_barrel_Lr0', 'em_barrel_Lr1_fine', 'em_barrel_Lr2' , 'em_barrel_Lr3',  
            'tile_barrel_Lr1', 'tile_barrel_Lr2', 'tile_barrel_Lr3', 'tracks_image']  
scalars   = ['p_Eratio', 'p_Reta' , 'p_Rhad' , 'p_Rphi' , 'p_TRTPID' , 'p_numberOfSCTHits' ,  
            'p_ndof' , 'p_dPOverP', 'p_deltaEta1', 'p_f1' , 'p_f3' , 'p_deltaPhiRescaled2',  
            'p_weta2' , 'p_d0' , 'p_d0Sig' , 'p_qd0Sig', 'p_nTracks', 'p_sct_weight_charge']  
others    = ['eventNumber', 'p_TruthType', 'p_iffTruth', 'p_LHTight', 'p_LHMedium', 'p_LHLoose',  
            'p_eta', 'p_et_calor', 'p_LHValue']
```

- DNNCaloSim (offline reconstruction, see related [talks](#))
- ML-based classification/calibration to topoclusters ([talks](#))
- Offline [CaloMuon ID](#) in athena/master.
- We are providing more example ([!38835](#)) to incorporate batch processing in onnxruntime.

Summary and outlook

- Currently onnxruntime has become no. 1 choice as ML framework for athena as it ticked off preliminary requirements
 - ✓ C++ API,
 - ✓ Supports complex models (e.g. CNN, RNN, GAN, VAE),
 - ✓ Thread exclusive processing
- However we believe we still need further tests to prove its robustness and longevity.
 - We invite people in atlas to incorporate onnxruntime in their ML related work; that is the way I believe we can know more and find its limitations and required areas of improvement.
- We tried to keep onnxruntime inference straight forward as possible but we definitely see a scope to make an abstraction of C++ API as an ML service (like other services in athena).