

CVN Integration into LArSoft

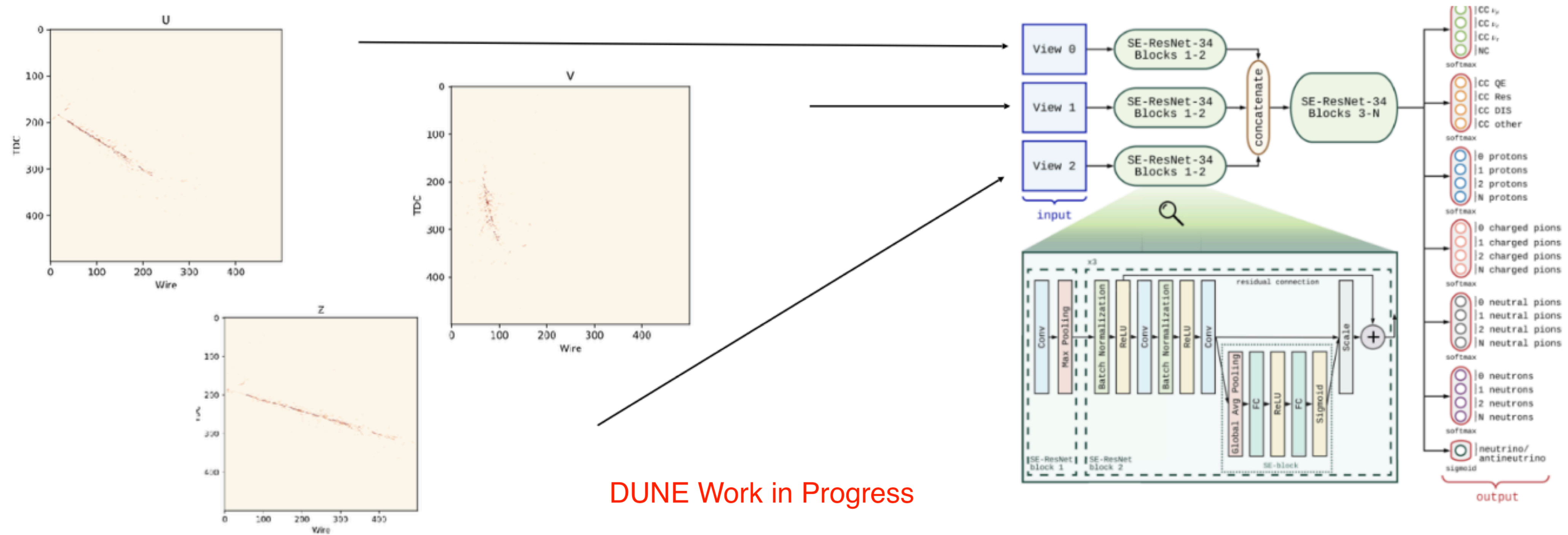
Nitish Nayak, Tingjun Yang, Varuna Meddage

18th Oct, 2022

Introduction

- Parts of dune reconstruction code, specifically some of the ML stuff could be beneficial to other LAr experiments (SBND, Argoneut, MicroBooNE etc)
 - And vice-versa
- Generally an active area of development on the DUNE side, but over time has grown crusty and could use an update
- The overall scheme is to have art modules to process simulated DUNE files (hits, clusters etc) — “pre-processing”
 - Some output format to be read by standalone python code — custom ML frameworks (not subject of this talk)
- More modules to read in trained results from Tensorflow, PyTorch and store them as new data products

CVN



DUNE Work in Progress

- CVN — highly performant neutrino ID developed for horizontal drift by Saul A.M and Leigh W. [<https://arxiv.org/abs/2006.15052>]
- Uses images of hit clusters (“pixel maps”) as input directly to the Convolutional Neural Network — no downstream reco necessary
 - Hits are gaussian-fits to deconvolved waveforms from the WireCell 2D signal processing chain (“gaushit”)
- Deep network architecture allows large performance gains not easily accessible through traditional methods
- Inputs are 2D Wire-Tick maps for each view and fed into a “Siamese Tower”-like architecture. Network merges information from the different views in the downstream layers

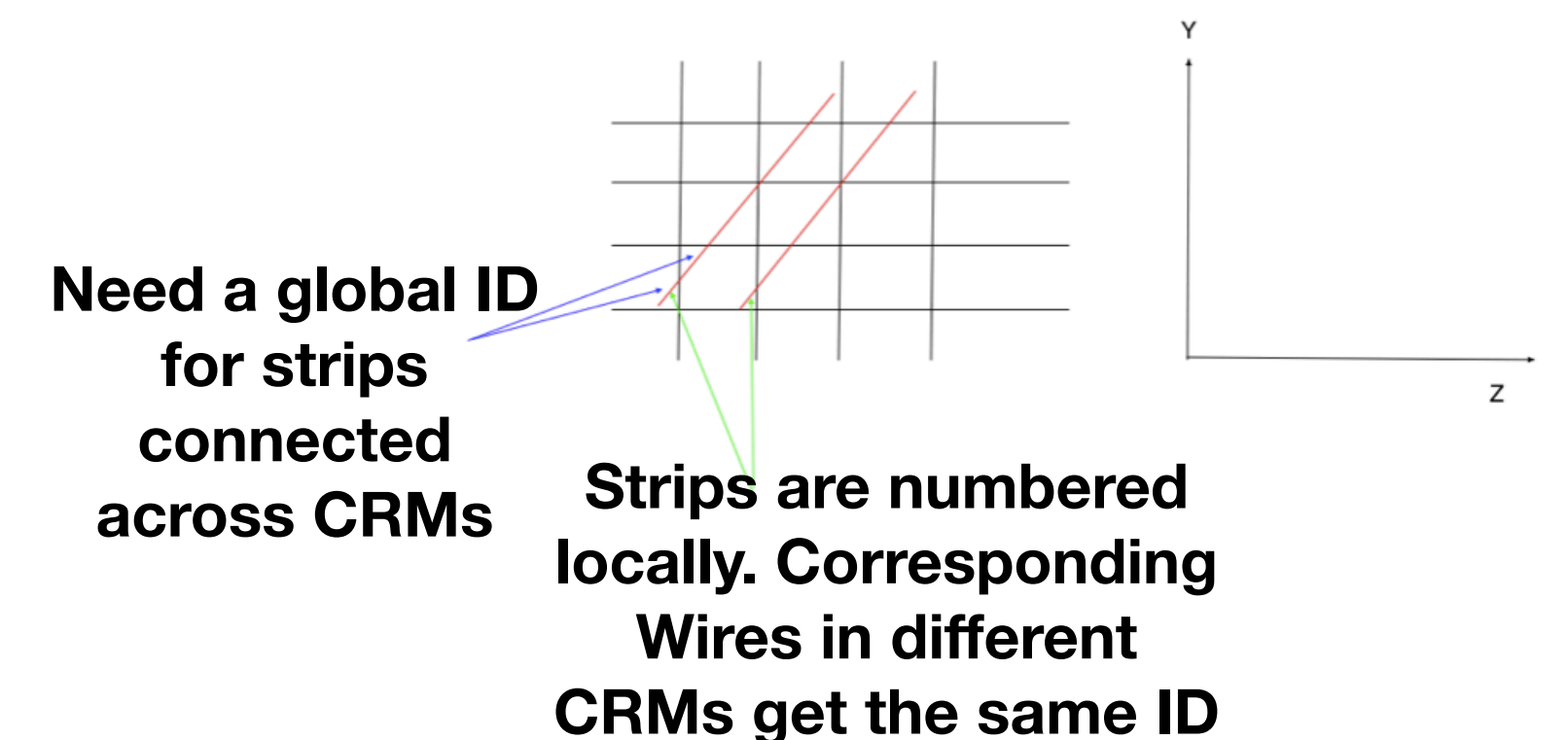
CVN in DUNE

```
29 // Producer algorithm for PixelMap, input to CVN neural net
30 class PixelMapProducer
31 {
32 public:
33   PixelMapProducer(unsigned int nWire, unsigned int nTdc, double tRes);
34   PixelMapProducer();
35
36   void SetUnwrapped(unsigned short unwrap){fUnwrapped = unwrap;};
37   void SetProtoDUNE(){fProtoDUNE = true;};
38
39   // Get boundaries for pixel map representation of cluster
40   Boundary DefineBoundary(detinfo::DetectorPropertiesData const& detProp,
41                           const std::vector< const recob::Hit* >& cluster);
42
43   // Function to convert to a global unwrapped wire number
44   void GetDUNEGlobalWire(unsigned int localWire, unsigned int plane, unsigned int tpc, unsigned int& globalWire, unsigned int& globalPlane) const;
45   void GetDUNEGlobalWireTDC(detinfo::DetectorPropertiesData const& detProp,
46                             unsigned int localWire, double localTDC, unsigned int plane, unsigned int tpc,
47                             unsigned int& globalWire, unsigned int& globalPlane, double& globalTDC) const;
48
49   void GetDUNE10ktGlobalWireTDC(detinfo::DetectorPropertiesData const& detProp,
50                                 unsigned int localWire, double localTDC, unsigned int plane, unsigned int tpc,
51                                 unsigned int& globalWire, unsigned int& globalPlane, double& globalTDC) const;
52   void GetProtoDUNEGlobalWire(unsigned int localWire, unsigned int plane, unsigned int tpc, unsigned int& globalWire, unsigned int& globalPlane) const;
53   void GetProtoDUNEGlobalWireTDC(unsigned int localWire, double localTDC, unsigned int plane, unsigned int tpc,
54                                   unsigned int& globalWire, double& globalTDC, unsigned int& globalPlane) const;
55   // preliminary vert drift 3 view studies
56   void GetDUNEVertDrift3ViewGlobalWire(unsigned int localWire, unsigned int plane, unsigned int tpc, unsigned int& globalWire, unsigned int& globalPlane) const;
57
58
```



- Main class that handles the creation of pixel map objects to be written out for training
- Essentially, takes in a cluster of hits -> sorts them into 2D wire-tick space for each view
- Need a global wire, tick co-ordinate for events spanning multiple APAs, drift vols, wrapped wires etc
- Handles different geometries within DUNE (HD, VD, ProtoDUNE-SP)
- Internally, some if-else conditions based on geometry service

GetDUNEVertDrift3ViewGlobalWire



CVN in DUNE

CVNMapper

```
113 //.....
114 void CVNMapper::produce(art::Event& evt)
115 {
116     // Use unwrapped pixel maps if requested
117     // 0 means no unwrap, 1 means unwrap in wire, 2 means unwrap in wire and time
118     fProducer.SetUnwrapped(fUnwrappedPixelMap);
119
120     std::vector< art::Ptr< recob::Hit > > hitlist;
121     auto hitListHandle = evt.getHandle< std::vector< recob::Hit > >(fHitsModuleLabel);
122     if (hitListHandle)
123         art::fill_ptr_vector(hitlist, hitListHandle);
124     unsigned short nhits = hitlist.size();
125
126     //Declaring containers for things to be stored in event
127     std::unique_ptr< std::vector<cvn::PixelMap> >
128         pmCol(new std::vector<cvn::PixelMap>);
129
130     if (nhits > fMinClusterHits) {
131         auto const detProp = art::ServiceHandle<detinfo::DetectorPropertiesService const>()->DataFor(evt);
132         PixelMap pm = fProducer.CreateMap(detProp, hitlist);
133         pm.SetTotHits(nhits);
134         pmCol->push_back(pm);
135     }
136     evt.put(std::move(pmCol), fClusterPMLabel);
137     //std::cout<<"Map Complete!"<<std::endl;
138 }
139
```

CVNZlibMaker

```
// cropped from 2880 x 500 to 500 x 500 here
std::vector<unsigned char> pixel_array(3 * fPlaneLimit * fTDCLimit);

CVNImageUtils image_utils(fPlaneLimit, fTDCLimit, 3);
image_utils.SetPixelMapSize(td.fPMap.NWire(), td.fPMap.NTdc());
image_utils.SetLogScale(fSetLog);
image_utils.SetViewReversal(fReverseViews);
image_utils.ConvertPixelMapToPixelArray(td.fPMap, pixel_array);

ulong src_len = 3 * fPlaneLimit * fTDCLimit; // pixelArray length
ulong dest_len = compressBound(src_len); // calculate size of the compressed data
char* ostream = (char *) malloc(dest_len); // allocate memory for the compressed data

int res = compress((Bytef *) ostream, &dest_len, (Bytef *) &pixel_array[0], src_len);
```

- Producer module (CVNMapper) uses PixelMapProducer to write out pixel map data products
- Analyzer module (CVNZlibMaker) uses these to write out pixel map objects as compressed binary files for each event
 - Also writes out some auxiliary information needed for training (flavor, neutrino energy etc)
- Problem arises when :
 - Want to write out pixel maps for not just recob::Hits but waveforms (recob::Wire) or even truth energy deposits (sim::SimChannel) directly. PixelMapProducer doesn't need to change a whole lot for this but not the way its currently set up
 - A lot of this code can also be made geometry agnostic, especially useful if one wants to adapt it for other LAr experiments (SBND, Argoneut etc) .

CVN in ArgoNeut

<https://cdcvns.fnal.gov/redmine/projects/argoneutcode/repository/revisions/develop/entry/CVNArgoneut/art/PixelMapProducer.h>

```
40 namespace cvn
41 {
42   // Producer algorithm for PixelMap, input to CVN neural net
43   class PixelMapProducer
44   {
45   public:
46     PixelMapProducer(unsigned int nWire, unsigned int nTdc, double tRes, unsigned int nPlanes, std::vector<bool> usePlaneVec, std::vector<double> calibrationConstants); // www: add nPlanes and usePlaneVec
47
48     void SetUnwrapped(unsigned short unwrap){fUnwrapped = unwrap;};
49     void SetProtoDUNE(){fProtoDUNE = true;};
50     void SetArgoNeuT(){fArgoNeuT = true;}; //www
51
52     // Get boundaries for pixel map representation of cluster
53     Boundary DefineBoundary(std::vector< art::Ptr< recob::Hit > >& cluster);
54
55     // function to convert to a global unwrapped wire number
56     void GetDUNEGlobalWire(unsigned int localWire, unsigned int plane, unsigned int tpc, unsigned int& globalWire, unsigned int& globalPlane) const;
57     void GetDUNEGlobalWireTDC(detinfo::DetectorPropertiesData const& detProp,
58                               unsigned int localWire, double localTDC, unsigned int plane, unsigned int tpc, unsigned int& globalWire, unsigned int& globalPlane, double& globalTDC) const;
59     void GetProtoDUNEGlobalWire(unsigned int localWire, unsigned int plane, unsigned int tpc, unsigned int& globalWire, unsigned int& globalPlane) const;
60     void GetArgoNeuTGlobalWire(unsigned int localWire, unsigned int plane, unsigned int tpc, unsigned int& globalWire, unsigned int& globalPlane) const; // www
61     void GetArgoNeuTGlobalWireTDC(unsigned int localWire, double localTDC, unsigned int plane, unsigned int tpc, unsigned int& globalWire, unsigned int& globalPlane, double& globalTDC) const; // www
62
63     unsigned int NWire() const {return fNWire;};
64     unsigned int NTdc() const {return fNTdc;};
65     double TRes() const {return fTRes;};
66     unsigned int NPlanes() const {return fNPlanes;}; //www
67     std::vector<double> CalibrationConstants() {return fCalibrationConstants;}; //www
68 }
```

Doesn't even belong here



- For eg, when CVN was adapted for Argoneut, the entire DUNE code was copied over to argoneutcode
- Few things were changed in PixelMapProducer for argoneut specific geometry, but otherwise lot of overlap
- Now SBND too -> not sustainable
- There exists a common repository — [larrecodnn](#)
 - Would be great if this could be ported over so that all experiments can write geometry specific code on top and use it
 - In the process, could also overhaul a lot of the framework to handle multiple inputs more rationally (recob::Hit, recob::Wire, calibrated Hits etc)

New CVN Framework

```
92 // Producer algorithm for PixelMap, input to CVN neural net
93 template <class T, class U> class PixelMapProducer
94 {
95 public:
96     PixelMapProducer(unsigned int nWire, unsigned int nTdc, double tRes, double threshold = 0.);
97     PixelMapProducer();
98
99     // overload constructor for inputs from fcl
100    PixelMapProducer(const fhicl::ParameterSet& pset);
101
102    void SetMultipleDrifts() {fMultipleDrifts = true;}
103    unsigned int NROI(){return fTotHits;};
104
105    // Get boundaries for pixel map representation of cluster
106    virtual Boundary DefineBoundary(detinfo::DetectorPropertiesData const& detProp,
107                                   const std::vector< const T* >& cluster);
108
109    virtual void ConvertLocaltoGlobal(geo::WireID wireid,
110                                     unsigned int &globalWire, unsigned int &globalPlane) const;
111
112    virtual void ConvertLocaltoGlobalTDC(geo::WireID wireid, double localTDC,
113                                       unsigned int &globalWire, unsigned int &globalPlane,
114                                       double &globalTDC) const;
```

```
typedef PixelMapProducer<recob::Hit, cvn::HitHelper> PixelMapHitProducer;
typedef PixelMapProducer<recob::Wire, cvn::WireHelper> PixelMapWireProducer;
typedef PixelMapProducer<sim::SimChannel, cvn::SimChannelHelper> PixelMapSimProducer;
```

```
32 Waveform HitHelper::GetWaveform()
33 {
34
35     Waveform ret;
36     double pe = fHit.Integral();
37     if (pe > fThreshold)
38         ret.push_back(std::map<double, double>({{fHit.PeakTime(), pe}}));
39
40     return ret;
41 }
42
43 geo::WireID HitHelper::GetID()
44 {
45     return fHit.WireID();
46 }
```

- Working out of a branch on a fork : https://github.com/nitish-nayak/larrecodnn/tree/feature/bnayak_cvncommon
- Make PixelMapProducer a class template
 - Can also write your own Helper class to access required information from the inputs (for example calibrated energy deposits vs raw energy deposits for recob::Hits)
 - Homogenize handling of other types of inputs (waveforms, truth energy deposits etc)
 - Geometry specific handling of wire, tick co-ordinates can be done by deriving from a base type and then overriding ConvertLocaltoGlobal or ConvertLocaltoGlobalTDC

New CVN Framework

```
35 namespace cvn {
36
37 template <class T, class U> class ICVNMapper : public art::EDProducer {
38 public:
39     explicit ICVNMapper(fhicl::ParameterSet const& pset);
40     ~ICVNMapper();
41
42     void produce(art::Event& evt);
43     void beginJob();
44     void endJob();
45
46 protected:
47     /// Module label for input clusters
48     std::string fHitsModuleLabel;
49
50     /// Instance label for cluster pixelmaps
51     std::string fClusterPMLabel;
52
53     /// Minimum number of hits for cluster to be converted to pixel map
54     unsigned short fMinClusterHits;
55
56     /// PixelMapProducer does the work for us
57     T fProducer;
58
59 };
60
61 }
```

```
4 #include "lardataobj/RecoBase/Hit.h"
5
6 #include "larrecodnn/CVN/module_helpers/PixelMapProducer.h"
7
8 namespace cvn {
9
10     typedef ICVNMapper<cvn::PixelMapHitProducer, recob::Hit> LArCVNHitMapper;
11     template class ICVNMapper<cvn::PixelMapHitProducer, recob::Hit>;
12
13 DEFINE_ART_MODULE(cvn::LArCVNHitMapper)
14 }
```

- Since the meat of the pixel map producer module is the PixelMapProducer class, we can similarly make a class template for the producer module
- Very simple task to make different variations of the producer modules for different inputs
 - Write your own experiment specific PixelMapProducer class (just involves deriving from original and overriding a function or two)
 - Writing new producer module with minimal code
 - For eg, PixelMapDUNEHitProducer w/ DUNECVNHitMapper etc

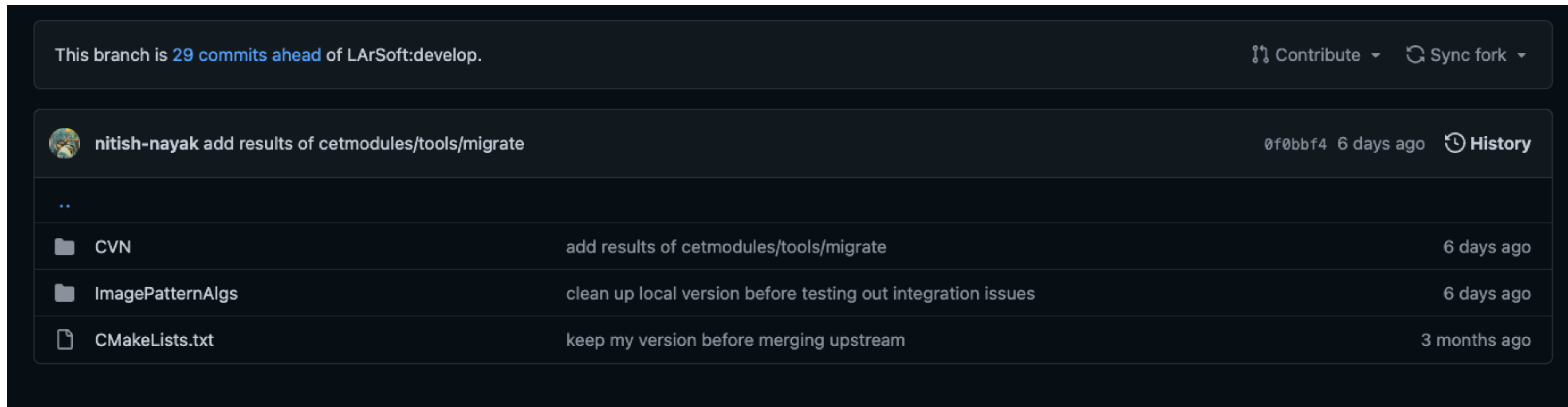
New CVN Framework

```
54 namespace cvn {
55
56     class ICVNZlibMaker : public art::EDAnalyzer {
57     public:
58
59         explicit ICVNZlibMaker(fhicl::ParameterSet const& pset);
60         ~ICVNZlibMaker();
61
62         void beginJob() override;
63         void analyze(const art::Event& evt) override {}
64         void reconfigure(const fhicl::ParameterSet& pset);
65
66     protected:
67
68         std::string fOutputDir;
69         std::string fPixelMapInput;
70         bool fSetLog;
71         std::vector<bool> fReverseViews;
72         unsigned int fPlaneLimit;
73         unsigned int fTDCLimit;
74
75         std::string out_dir;
76         CVNImageUtils fImage;
77
78         template <class T> void write_files(TrainingData<T> td, std::string evtid) = delete;
79
80     };
```


- Similarly for writing out compressed binaries
- Improve how auxiliary information is also written out, so the user can just have a much cleaner interface for all this
- I think these are all mostly long overdue changes which prevent unnecessary complexity for many different variations people want to try out
- Not meant to be a complete rethink on how to do the pre-processing in general

Current Status




- Varuna has been using my fork to use it on SBND
 - It works, able to produce pixel maps and do the training
- Want to submit a PR and integrate it into larrecodnn
 - This had been on hold for a while when Varuna was testing it on SBND - my mistake
- Unfortunately, in the meantime a lot of things have changed on the LArSoft side
 - Tensorflow version (easily adapted)
 - Cetmodules migration from v2 -> v3



This branch is [29 commits ahead](#) of LArSoft:develop. [Contribute](#) [Sync fork](#)

 **nitish-nayak** add results of cetmodules/tools/migrate 0f0bbf4 6 days ago [History](#)

..

 CVN	add results of cetmodules/tools/migrate	6 days ago
 ImagePatternAlgs	clean up local version before testing out integration issues	6 days ago
 CMakeLists.txt	keep my version before merging upstream	3 months ago

Cetmodules migration

Chris Green's talk

- “What’s the difference between building with Cetbuildtool 8 and building for Cetmodules 2?”
 - Short answer: “Not very much at all, really.”
 - Long answer: “Pretty much everything, really.”

```
... @@ -1,30 +1,150 @@
1 - art_make( MODULE_LIBRARIES
2 -   larreco_RecoAlg_ImagePatternAlgs_DataProvider
3 -   lardata_ArtDataHelper
4 +   lardataobj_RawData
5 -   art::Framework_Services_System_TriggerNamesService_service
6 -   larsim_MCcheater_ParticleInventoryService_service
7 -   larreco_Calorimetry
8 -   larcorealg_Geometry
9 -   lardataalg_DetectorInfo
10 -   lardataobj_RecoBase
11 -   nusimdata::SimulationBase
12 -   nurandom::RandomUtils_NuRandomService_service
13 -   art::Framework_Core
14 -   art::Framework_Principal
15 -   art::Framework_Services_Registry
16 -   art_root_io::tfile_support
17 -   art_root_io::TfileService_service
18 -   art::Framework_Services_Optional_RandomNumberGenerator_service
19 -   art::Persistency_Common
20 -   art::Persistency_Provenance
21 -   art::Utilities
22 -   canvas::canvas
23 -   messagefacility::MF_MessageLogger
24 -   CLHEP::Random
25 -   ROOT::Core
26 -   ROOT::Tree
27 - ) ## MIGRATE-ACTION-RECOMMENDED (migrate-3.16.00) - deprecated: use art_make_library(), art_dictionary(), and
   cet_build_plugin() with explicit source lists and plugin base types
1 + cet_make_library(LIBRARY_NAME EmTrack INTERFACE
2 + SOURCE
3 + EmTrack.h
4 + LIBRARIES INTERFACE
5 + larrecodnn::PointIdAlgorithm
6 + lardata::ArtDataHelper
7 + lardata::DetectorClocksService
8 + lardata::DetectorPropertiesService
9 + lardata::AssociationUtil
10 + lardataobj::RecoBase
11 + larcreeobj::SimpleTypesAndConstants
12 + art_plugin_support::toolMaker
13 + art::Framework_Core
14 + art::Framework_Services_Registry
15 + art::Framework_Services_System_TriggerNamesService_service
16 + canvas::canvas
17 + messagefacility::MF_MessageLogger
18 + fhiclcpp::types
19 + cetlib::container_algorithms
20 + cetlib_except::cetlib_except
21 + )
22 +
23 + cet_build_plugin(CheckQWNScore art::EDAnalyzer
24 + LIBRARIES PRIVATE
25 + lardata::ArtDataHelper
26 + lardataobj::RecoBase
27 + art_root_io::TfileService_service
28 + art::Framework_Services_Registry
29 + canvas::canvas
30 + fhiclcpp::fhiclcpp
31 + ROOT::Tree
32 + )
33 +
```

- My CMake config files were based on now deprecated methods
 - Uses art_make, doesn't use cet_make_library
 - Doesn't use explicit source lists for each plugin
- Overall, a lot of changes (some described [here](#))
- These changes have already been integrated into the rest of larrecodnn for other ML algorithms, under ImagePatternAlgs (See : [commit](#) by Chris Green)
- Currently don't have the knowledge/expertise to adapt to the new regime
- Created larsoft issue here : <https://cdcv.sfnal.gov/redmine/issues/27477>

Summary

- Successfully ported over most of the DUNE CVN code to larrecodnn
 - Experiment agnostic framework
 - Common interface for multiple types of input for pixel map production as well as output format
- Tested on SBND and it works, able to train and generate results
- To submit a PR :
 - Need to merge changes after migrating to new version of cetmodules
 - Need new CMake config files
- Wanted to ask advice on how to go about this
 - Maybe some support to implement the necessary changes as was done for other code in larrecodnn?
 - Any documentation if it exists that I can use to do this myself?

Thank you for listening!