

Real-time AI Triggering for Liquid Argon Time Projection Chambers

Seokju Chung (Columbia University)

CPAD 2024

November 21st, 2024

Real-Time Anomaly Detection

- [NSF-funded](#) collaborative project between Columbia and Princeton Universities
- Columbia (Neutrino) - G. Karagiorgi (PI), S. Chung, J. Cleeve, A. Malige
- Princeton (Collider) - I. Ojalvo (PI), L. Gerlach, Andrew Ji, A. Pol (now at Thomson Reuters Lab)

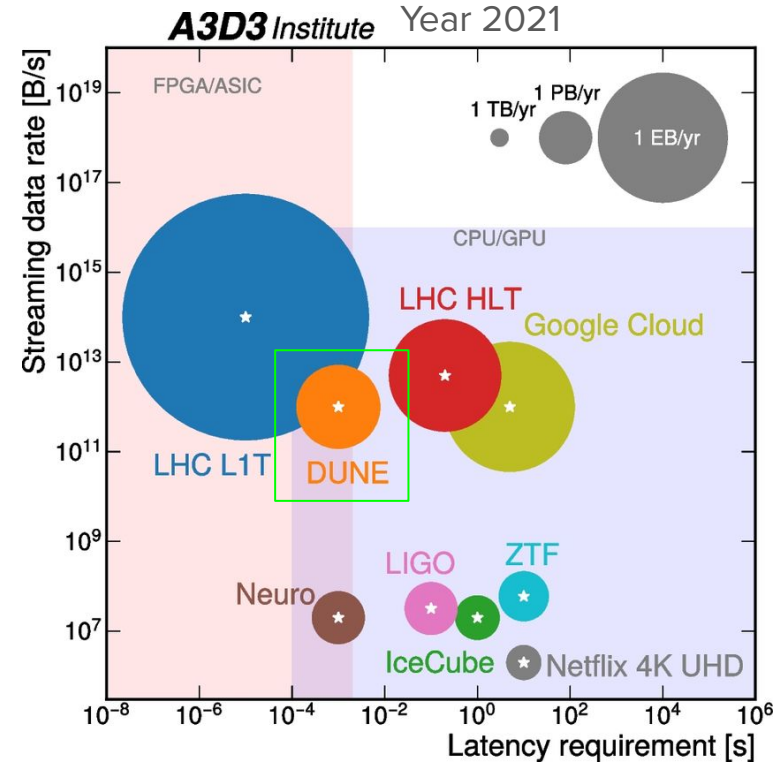


Award Abstract # 2209917

Elements: RAD Discoveries for Fundamental Physics

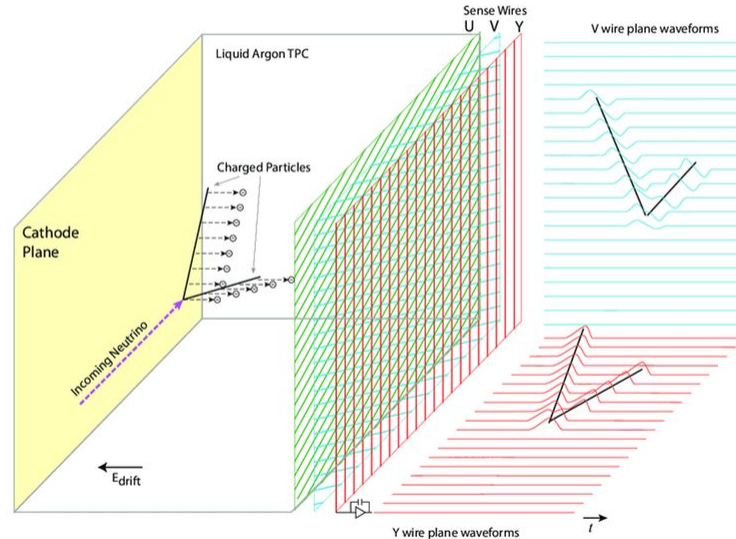
Why Real-time Triggering?

- Particle experiments generate large amounts of data
- Impossible to save all
 - Store in temporary buffer
 - Need selection (trigger) which decides whether to keep buffer data or not



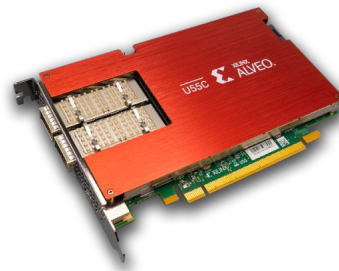
Liquid Argon Time Projection Chambers (LArTPCs)

- Widely used technology for neutrino physics (ArgoNeuT, MicroBooNE, SBND, ICARUS, DUNE, etc.)
- Neutrino interacts with Ar nucleus, creating charged particles
- Charged particles create ionization electrons, which are drifted in a large uniform electric field and sensed by wire sensor arrays



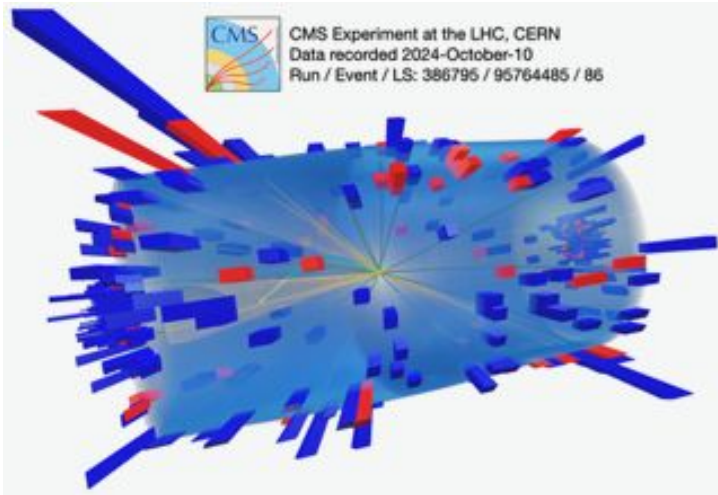
Hardware AI Trigger (Data Selection)

- Image processing: Analysis of image to identify ones that are of interest
- Input image processing rate needs to be faster than (generated) image streaming rate
- Could benefit from hardware acceleration
 - use Field Programmable Gate Array (FPGA)
 - DUNE: ~100 images/ms, ~14 Mpixel/image



Hardware AI Trigger

- Hardware AI triggers already being investigated/proposed for different particle experiments



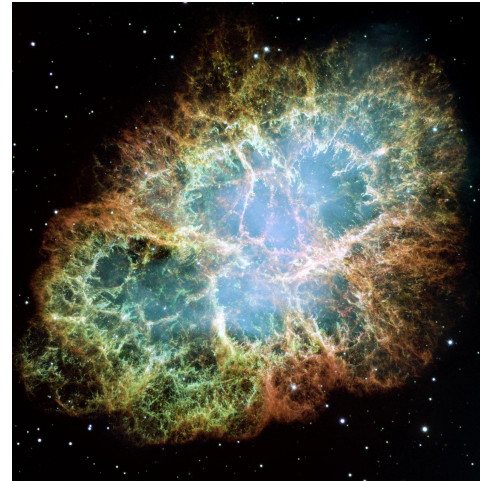
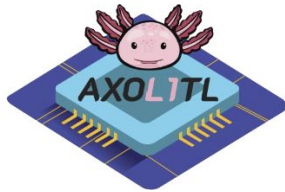
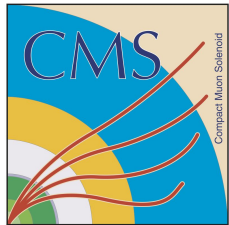
CMS Collaboration. **Model-Independent Real-Time Anomaly Detection at the CMS Level-1 Calorimeter Trigger with CICADA.** 2024

DP-2024/121

Yeon-jae Jwa, Giuseppe Di Guglielmo, Lukas Arnold, Luca Carloni, Georgia Karagiorgi. **Real-time Inference with 2D Convolutional Neural Networks on Field Programmable Gate Arrays for High-rate Particle Imaging Detectors.** *Front.Artif.Intell.* 5, 14 Jan 2022

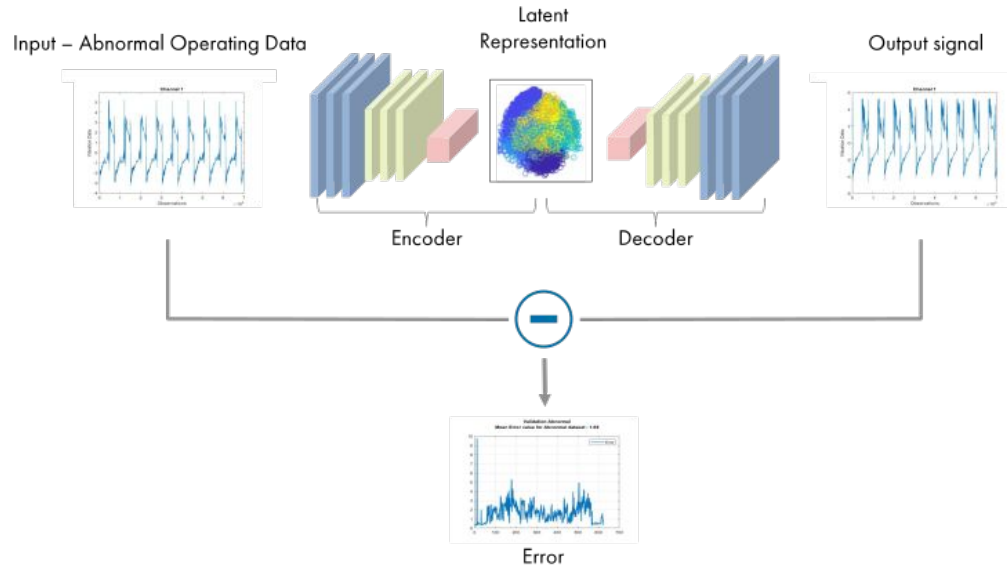
Why Trigger on Anomalies? - Anomaly Trigger

- Anomalies: Rarely occurring signatures (ex: Supernova, BSM)
- Conventional triggers designed based on expected particle signature
 - Model dependent; signature for new physics is unknown
- Model-independent trigger, should learning from data itself



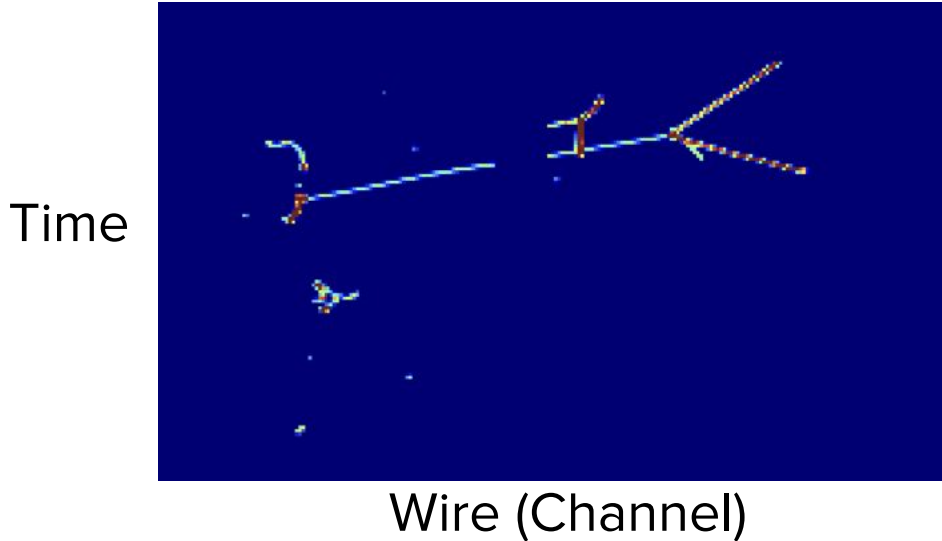
Detecting Anomalies

- Utilize **Autoencoders**, which encodes the data into a more compact format, and tries to reproduce original input by decoding it
- Autoencoder learns common features in data through unsupervised learning
- “Anomalous” events will have a larger difference between input and output
- Difference quantified as **Anomaly Score**



LArTPCs - Input Data

- Combine wire information (sensed ionization charge) as a function of time to get “image” of particle trajectories
- Use MicroBooNE Public Dataset for proof of concept on Autoencoders



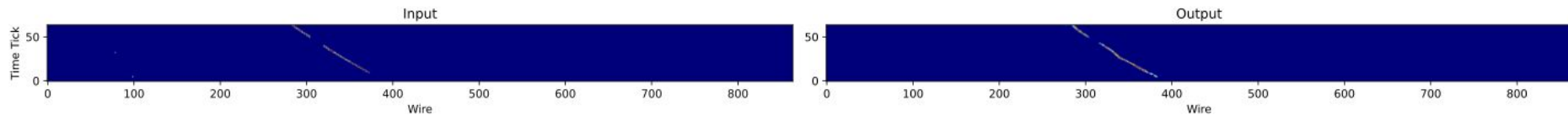
Made with MicroBooNE Public Dataset
[10.5281/zenodo.7262009](https://zenodo.org/record/7262009)

Network Performance

- See correlation between **Anomaly Score** and **Number of Tracks** in a given input image to the Autoencoder

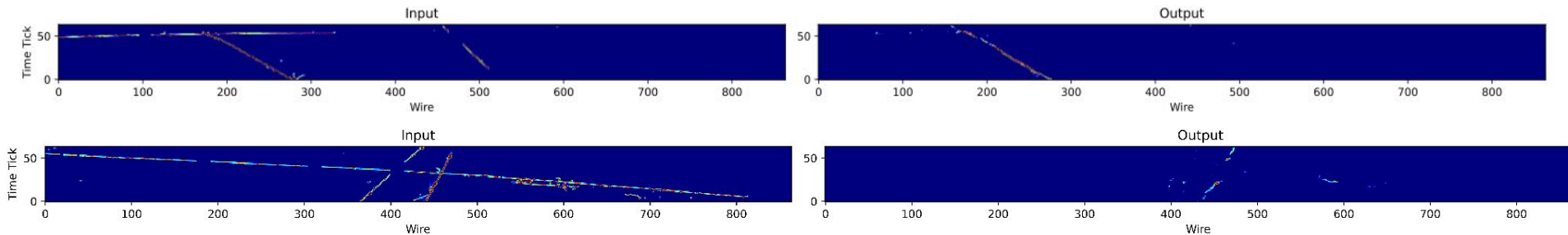
Preliminary

Low Anomaly Score Example



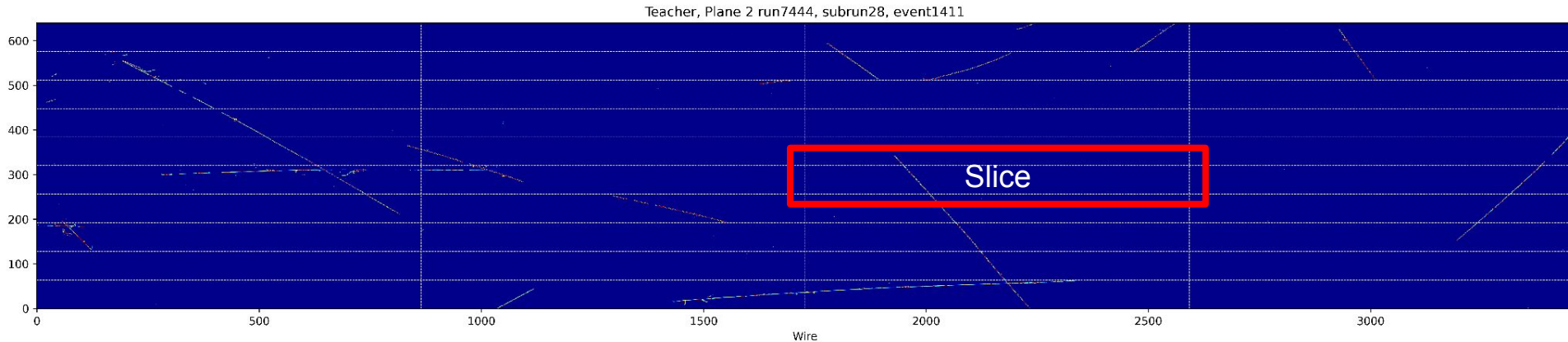
Preliminary

High Anomaly Score Example



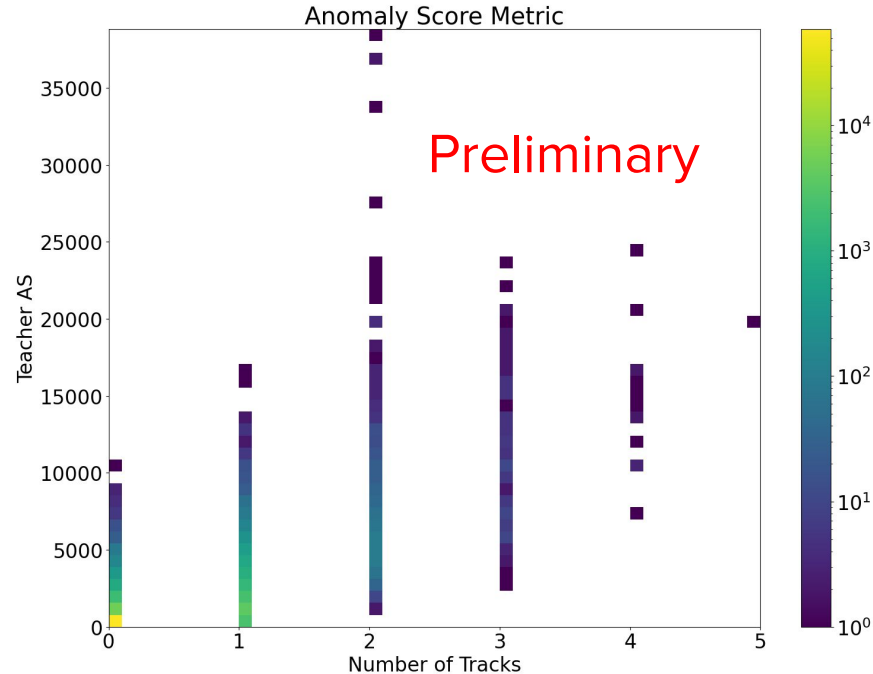
Network Performance

- Common feature: Empty or single track
- Anomaly expected to correlate with having multiple tracks
 - Expect higher Anomaly Scores with slices having multiple tracks



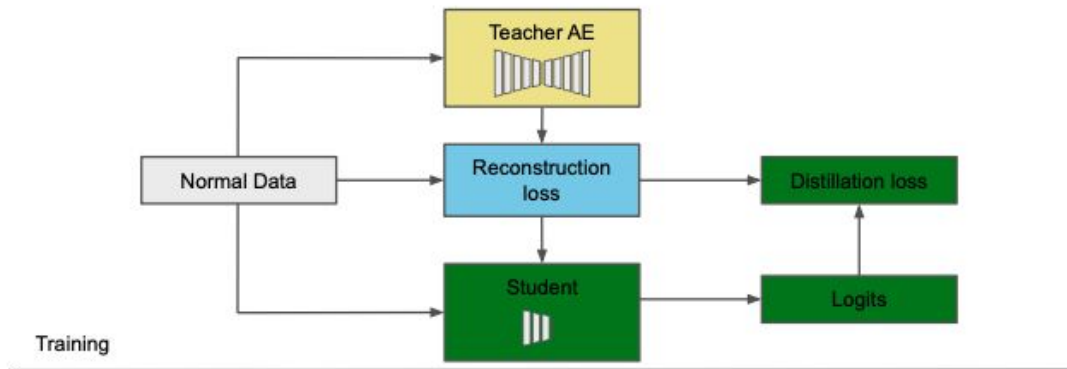
Network Performance

- See correlation between **Anomaly Score** and **Number of Tracks** in a given input image to the Autoencoder



Triggering with AI Algorithms

- Neural networks are effective; but, typically, their performance comes with a large computational and power resource consumption
- Using **Knowledge Distillation**, we project the performance of a (large, resource-intensive) Teacher Autoencoder to smaller **Student** quantized network, which can be deployed on power-efficient devices



Adrian Alan Pol, Ekaterina Govorkova, Sonja Gronroos, Nadezda Chernyavskaya, Philip Harris et al.
Knowledge Distillation for Anomaly Detection. Oct 9, 2023.

Triggering with AI Algorithms

- Size reduction by factor of ~ 75 (250 MB \rightarrow 3.4 MB)
- Teacher and Student Anomaly Scores are correlated
 - Smaller Student network maintains sensitivity to anomalies

```
Model: "teacher"
```

Layer (type)	Output Shape	Param #
teacher_inputs_ (InputLayer)	[(None, 864, 64, 1)]	0
teacher_reshape (Reshape)	(None, 864, 64, 1)	0
teacher_conv2d_1 (Conv2D)	(None, 864, 64, 20)	200
teacher_relu_1 (Activation)	(None, 864, 64, 20)	0
teacher_pool_1 (AveragePooling2D)	(None, 432, 32, 20)	0
teacher_conv2d_2 (Conv2D)	(None, 432, 32, 30)	5430
teacher_relu_2 (Activation)	(None, 432, 32, 30)	0
teacher_flatten (Flatten)	(None, 414720)	0
teacher_latent (Dense)	(None, 80)	33177680

Total params: 66789361 (254.78 MB)
Trainable params: 66789361 (254.78 MB)
Non-trainable params: 0 (0.00 Byte)

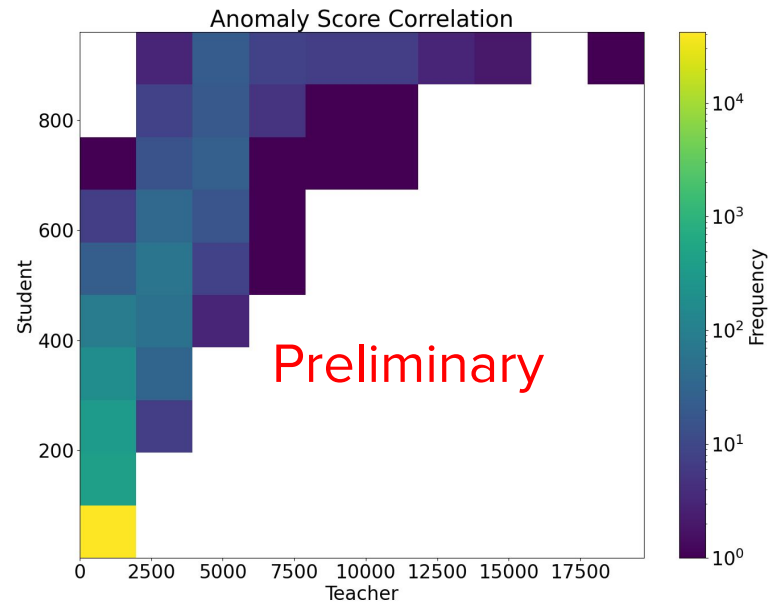
Teacher
(Larger)

```
Model: "v1_16X12"
```

Layer (type)	Output Shape	Param #
inputs_ (InputLayer)	[(None, 55296)]	0
dense1 (QDenseBatchnorm)	(None, 16)	884817
relu1 (QActivation)	(None, 16)	0
dropout_1 (Dropout)	(None, 16)	0
dense2 (QDense)	(None, 1)	16
outputs (QActivation)	(None, 1)	0

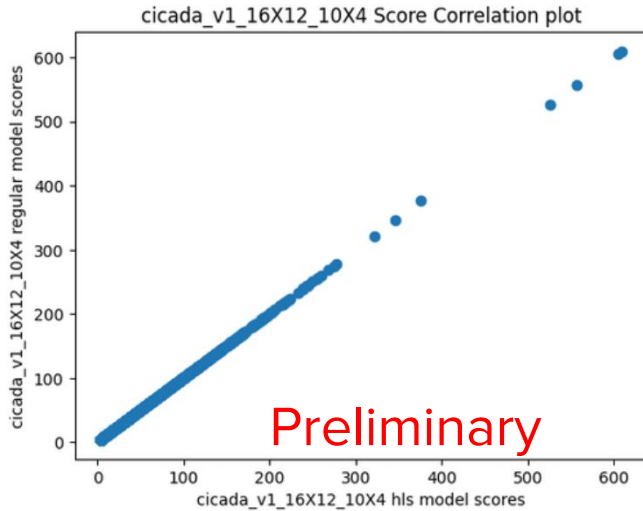
Total params: 884833 (3.38 MB)
Trainable params: 884800 (3.38 MB)
Non-trainable params: 33 (136.00 Byte)

Student
(Smaller)



Implementing the Network onto Hardware

- Trained QKeras network is converted using hls4ml
- Converted network written onto FPGA using either Vitis or Vivado platforms



Vitis HLS 2023.1 - myproject_prj (/mnt/storage1/jhenry/anomaly_test/work_dir/io_stream_v1_synth_test_reuse_8/myproject_prj)

Synthesis Summary(solution1) x myproject.cpp

Synthesis Summary Report of 'myproject'

General Information

Date:	Fri Sep 6 14:57:16 2024	Solution:	solution1 (Vitis Kernel Flow Target)
Version:	2023.1 (Build 3854077 on May 4 2023)	Product family:	virtexplus
Project:	myproject_prj	Target device:	xcvu13p-fga2577-2-e

Timing Estimate

Target	Estimated	Uncertainty
5.00 ns	4.369 ns	0.62 ns

Performance & Resource Estimates

Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Its
myproject	-	-	-	-	16	80.000	
dense_array_ap_fixed_288u_array_ap_fixed_32_16_5_3_0_16u_config2_s	-	-	-	-	8	40.000	
normalize_array_ap_fixed_16u_array_ap_fixed_32_16_5_3_0_16u_config4_s	-	-	-	-	2	10.000	
relu_array_ap_fixed_16u_array_ap_ufixed_10_6_4_0_0_16u_relu_config5_s	-	-	-	-	1	5.000	
dense_array_ap_ufixed_16u_array_ap_fixed_32_16_5_3_0_1u_config6_s	-	-	-	-	2	10.000	

Preliminary

Applications - Millicharged Particles

- Hypothetical BSM particles with fractional electron charge, generated from meson decays
- mCP tracks will have lower 'faint' ADC values compared to ordinary, etc. muons
- Benchmark BSM model → should be able to test in neutrino experiments (ex: SBND)

$$q = e$$

$$q = 0.1e$$

Conclusion

- Real-time AI triggers can be developed using an Autoencoder with Knowledge Distillation
- We are exploring its applicability to LArTPC data using the MicroBooNE Public Dataset; hardware implementation in progress
- Not limited to LArTPC only, adaptable to different input datasets and applications across HEP

This work was supported by the National Science Foundation under Grant No. OAC-2209917.

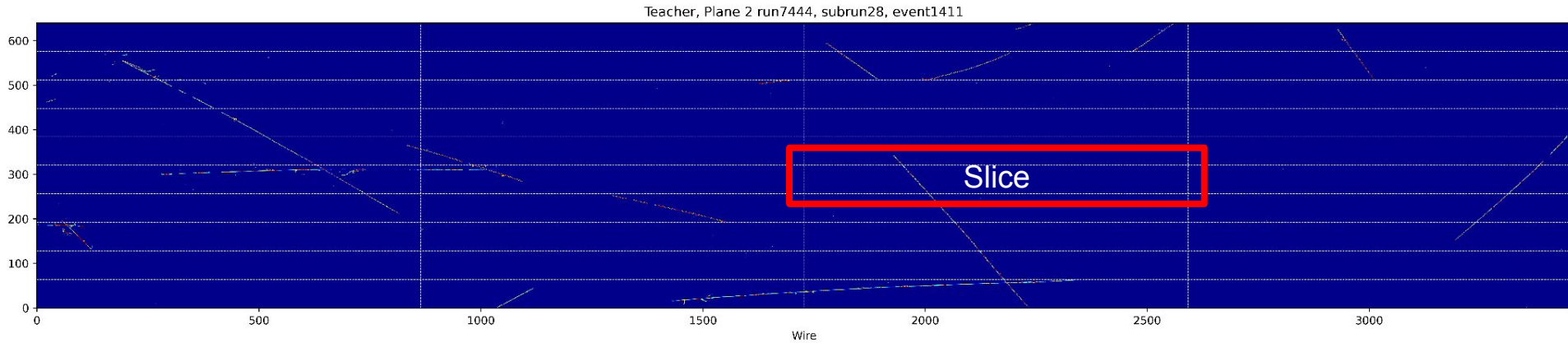
We acknowledge the MicroBooNE Collaboration for making publicly available the data sets [[10.5281/zenodo.7262009](https://doi.org/10.5281/zenodo.7262009)] employed in this work. These data sets consist of simulated neutrino interactions from the Booster Neutrino Beamline overlaid on top of cosmic data collected with the MicroBooNE detector [2017 JINST 12 P02017].

Thank you

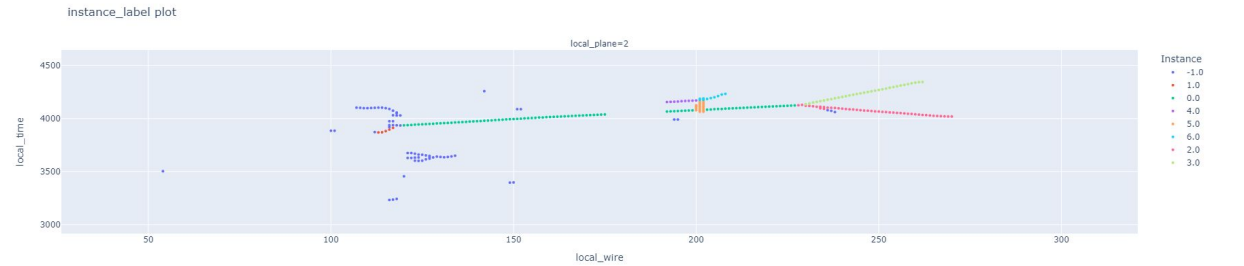
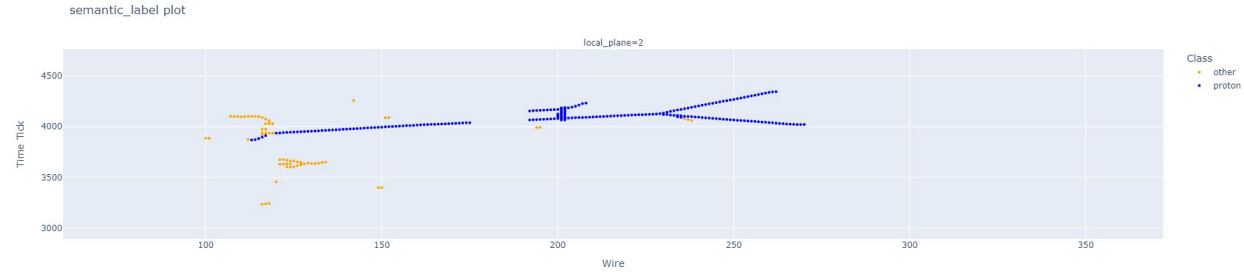
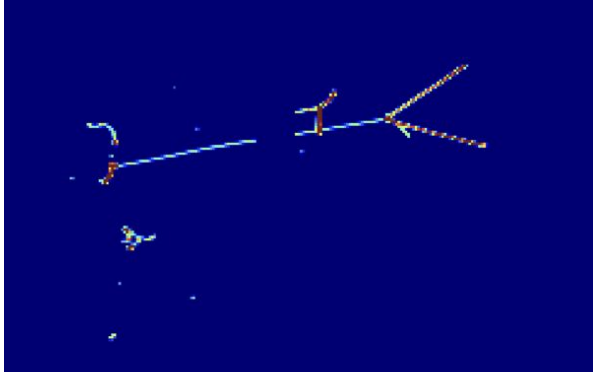
Backup

Input Data

- 3456 Wires X 6400 Time Ticks \rightarrow 3456 X 640 \rightarrow 864 X 64



Input Data



Network Structure

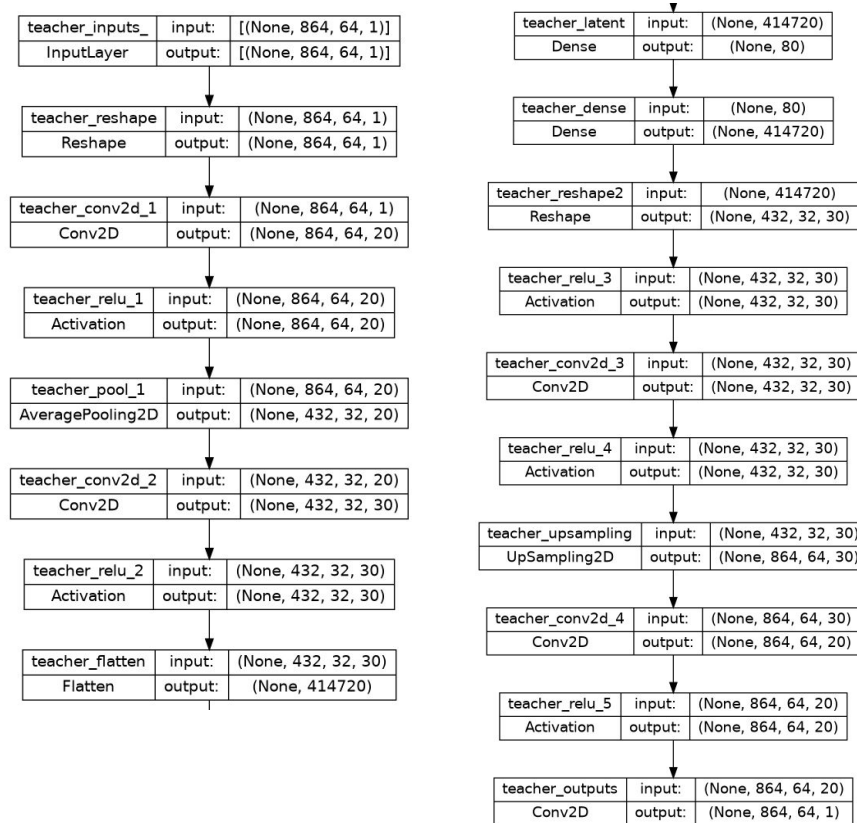
```
class TeacherAutoencoder:
    def __init__(self, input_shape: tuple):
        self.input_shape = input_shape

    def get_model(self):
        inputs = Input(shape=self.input_shape, name="teacher_inputs_")
        x = Reshape((864, 64, 1), name="teacher_reshape")(inputs)
        x = Conv2D(20, (3, 3), strides=1, padding="same", name="teacher_conv2d_1")(x)
        x = Activation("relu", name="teacher_relu_1")(x)
        x = AveragePooling2D((2, 2), name="teacher_pool_1")(x)
        x = Conv2D(30, (3, 3), strides=1, padding="same", name="teacher_conv2d_2")(x)
        x = Activation("relu", name="teacher_relu_2")(x)
        x = Flatten(name="teacher_flatten")(x)
        x = Dense(80, activation="relu", name="teacher_latent")(x)
        x = Dense(432 * 32 * 30, name="teacher_dense")(x)
        x = Reshape((432, 32, 30), name="teacher_reshape2")(x)
        x = Activation("relu", name="teacher_relu_3")(x)
        x = Conv2D(30, (3, 3), strides=1, padding="same", name="teacher_conv2d_3")(x)
        x = Activation("relu", name="teacher_relu_4")(x)
        x = UpSampling2D((2, 2), name="teacher_upsampling")(x)
        x = Conv2D(20, (3, 3), strides=1, padding="same", name="teacher_conv2d_4")(x)
        x = Activation("relu", name="teacher_relu_5")(x)
        outputs = Conv2D(
            1,
            (3, 3),
            activation="relu",
            strides=1,
            padding="same",
            name="teacher_outputs",
        )(x)
        return Model(inputs, outputs, name="teacher")
```

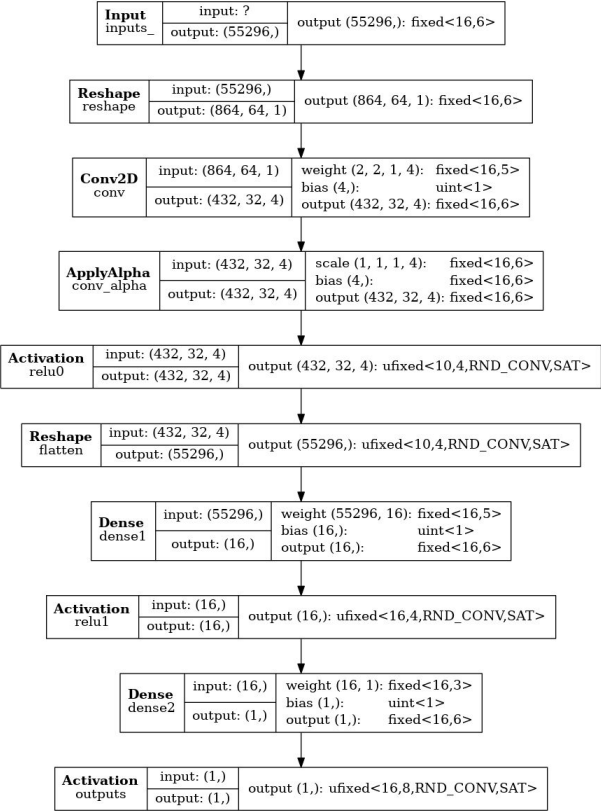
```
class V1_16X16:
    def __init__(self, input_shape: tuple):
        self.input_shape = input_shape

    def get_model(self):
        inputs = Input(shape=self.input_shape, name="inputs_")
        x = QDenseBatchnorm(
            16,
            kernel_quantizer=quantized_bits(16, 4, 1, alpha=1.0),
            bias_quantizer=quantized_bits(8, 3, 1, alpha=1.0),
            name="dense1",
        )(inputs)
        x = QActivation("quantized_relu(10, 6)", name="relu1")(x)
        x = Dropout(1 / 8)(x)
        x = QDense(
            1,
            kernel_quantizer=quantized_bits(12, 3, 1, alpha=1.0),
            use_bias=False,
            name="dense2",
        )(x)
        outputs = QActivation("quantized_relu(16, 8)", name="outputs")(x)
        return Model(inputs, outputs, name="v1_16X16")
```

Network Structure - Teacher



Network Structure - Student



Track Definition

- Track_{n_i}: [$n_i \leq$ numbers of particles $< n_{(i+1)}$] of the same type need to be inside the input image to be recognized as a single track

