A Machine Learning Framework for Gravitational Wave Signal Detection

Ethan Marx, Alec Gunny, Will Benoit, Rafia Omer, Eric Moreno, Ryan Raikman, Dylan Rankin, Philip Harris, Michael Coughlin, Erik Katsavounidis

Accelerating Physics with ML Conference, 1/30/23

Background

- LIGO is preparing for its upcoming 4th observing run
- Expect to detect ~1 CBC event per day
- Fast identification of sources important for informing EM follow up

Motivation: Benefits of ML

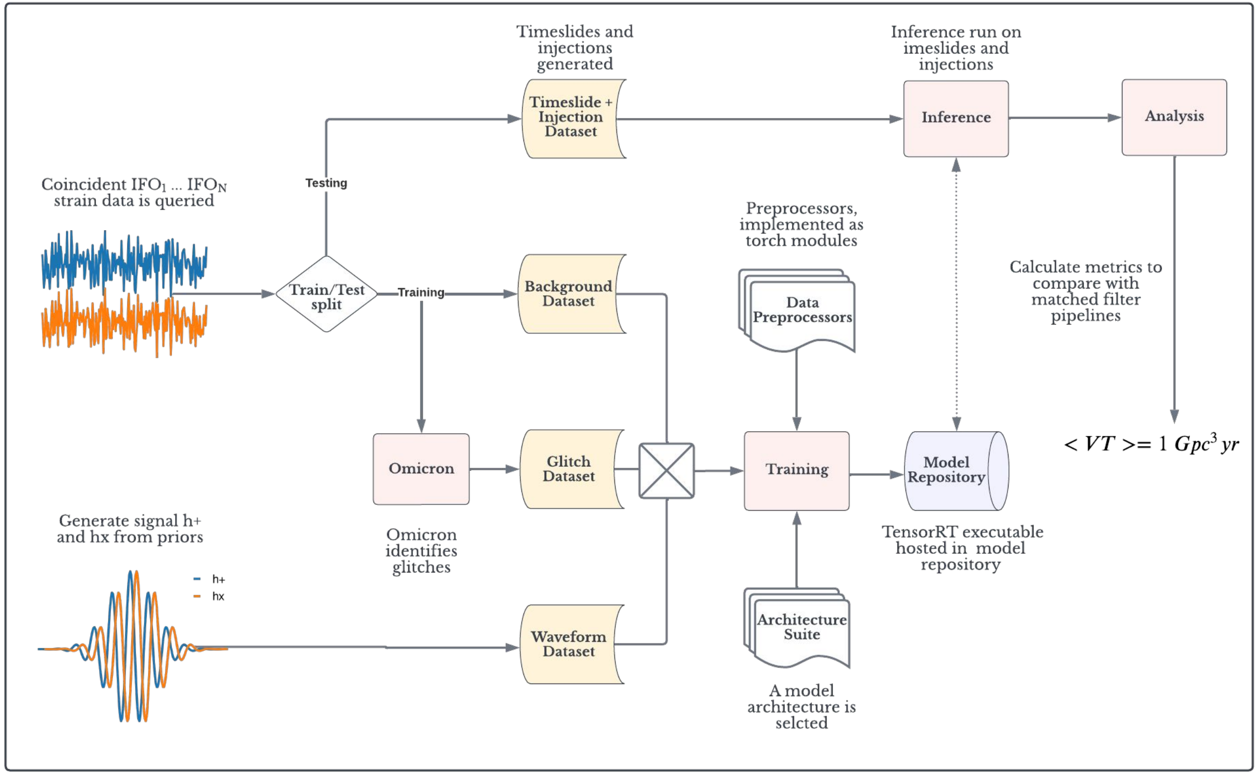
Online: Decreased inference latency for electromagnetic follow-up

Offline: Smaller computational footprint, larger throughput

Both: Template bank scalability, Increased Sensitivity?

BBHnet

Framework for optimizing neural networks to detect gravitational waves from time domain strain



Timeslides and injections generated

Inference run on imeslides and injections

Timeslide + Injection Dataset

Inference

Analysis

Coincident IFO_1 ... IFO_N strain data is queried

Testing

Preprocessors, implemented as torch modules

Calculate metrics to compare with matched filter pipelines

Train/Test split

Training

Background Dataset

Data Preprocessors

Omicron

Glitch Dataset

Training

Model Repository

$< VT >= 1\ Gpc^3\ yr$

Generate signal h+ and hx from priors

Omicron identifies glitches

TensorRT executable hosted in model repository

h+

hx

Waveform Dataset

Architecture Suite

A model architecture is selcted

ml4gw: Data Processing on GPU

Randomly slice kernels from background strain data

```
from ml4gw.utils.slicing import slice_kernels
```

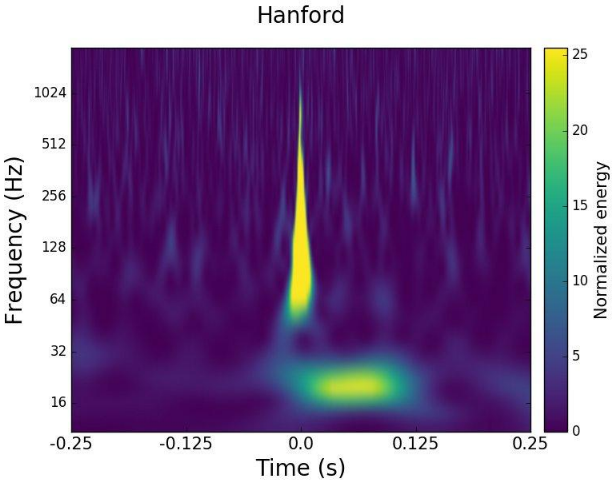
Waveform projection at training time

```
from ml4gw.transforms.injection import RandomWaveformInjection
```

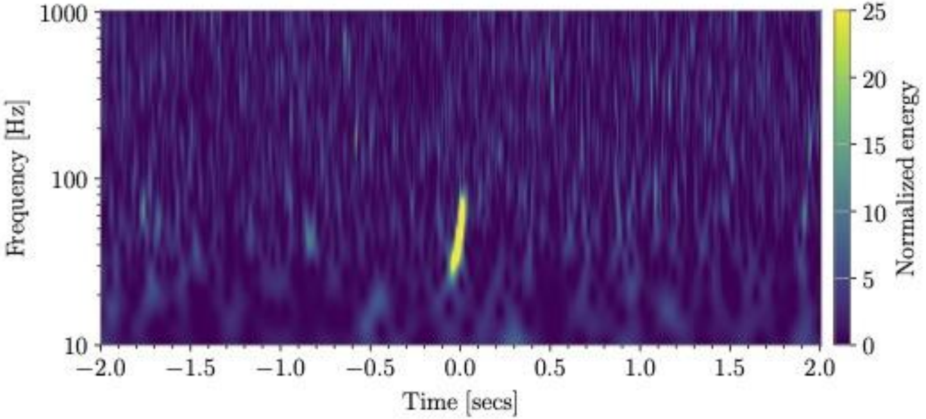
Whitening filter

```
from ml4gw.transforms.whitening import Whitening
```

Oversampling Glitches



Example Blip glitch in Hanford data

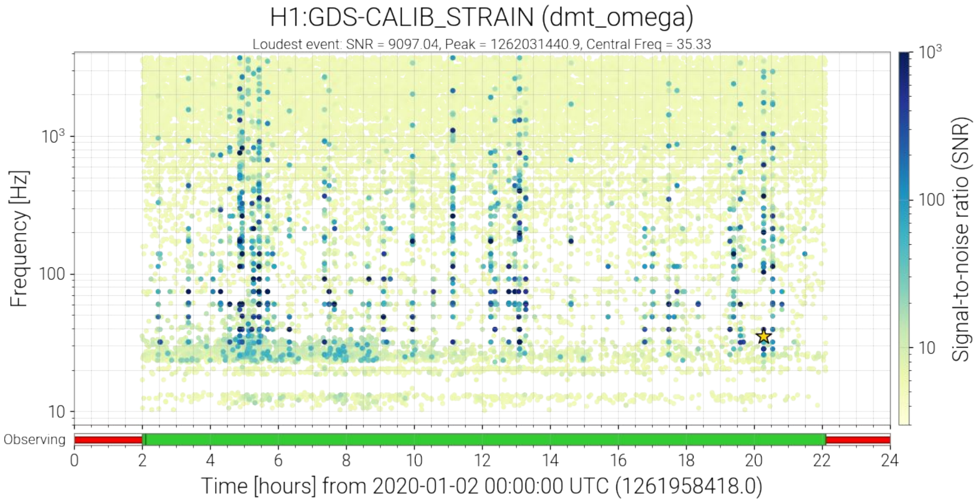


Example binary black hole merger

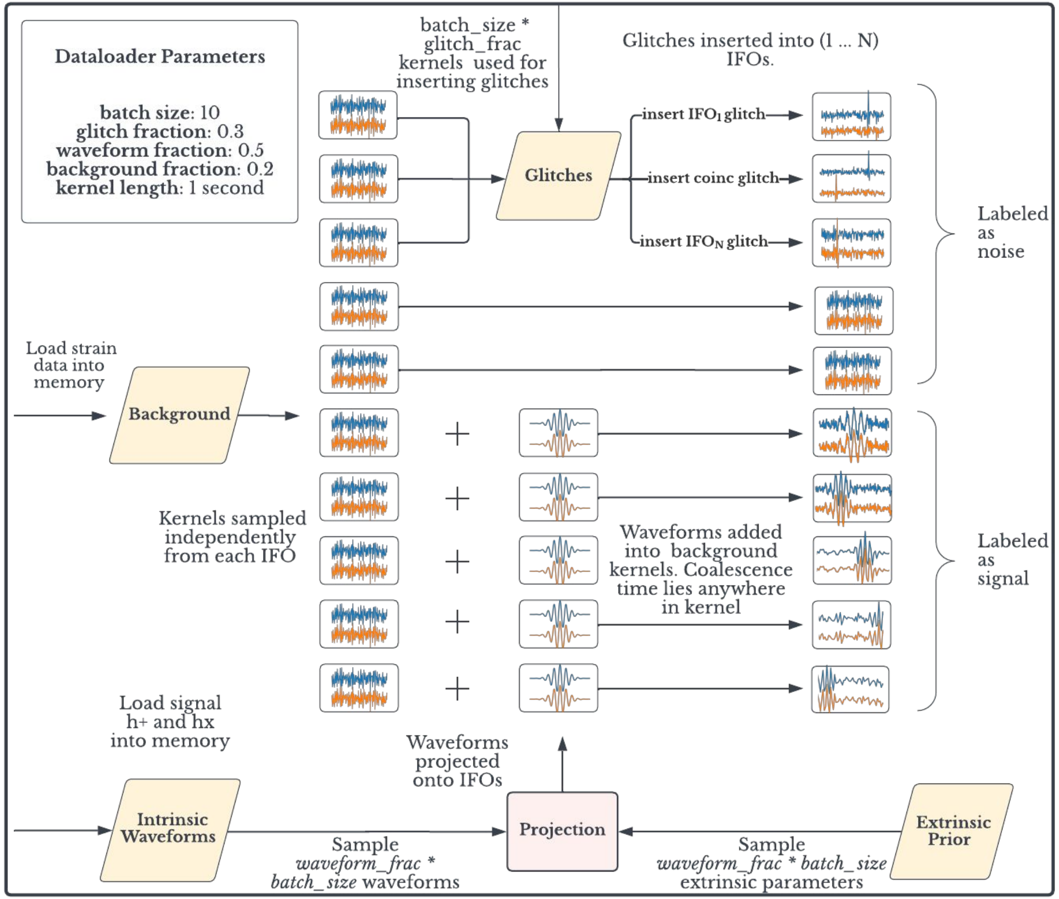
Oversampling Glitches

Utilize Omicron algorithm to identify “triggers” of excess power

Create dataset of glitch events from these triggers



Dataloader



Dataloader Parameters

batch size: 10
glitch fraction: 0.3
waveform fraction: 0.5
background fraction: 0.2
kernel length: 1 second

batch_size * glitch_frac kernels used for inserting glitches

Glitches inserted into (1 ... N) IFOs.

Glitches

insert IFO$_1$ glitch

insert coinc glitch

insert IFO$_N$ glitch

Labeled as noise

Load strain data into memory

Background

Kernels sampled independently from each IFO

Waveforms added into background kernels. Coalescence time lies anywhere in kernel

Labeled as signal

Load signal h+ and hx into memory

Intrinsic Waveforms

Sample *waveform_frac* * *batch_size* waveforms

Waveforms projected onto IFOs

Projection

Sample *waveform_frac* * *batch_size* extrinsic parameters

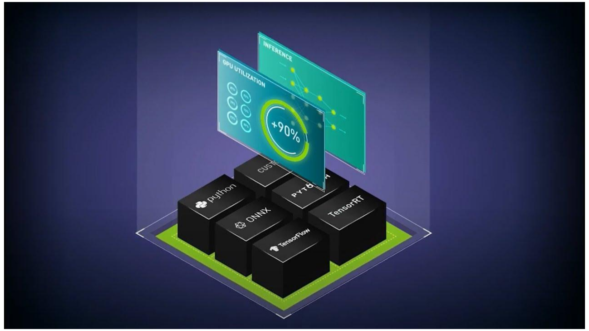
Extrinsic Prior

Model Evaluation

Model performance on signals evaluated through injections into background strain

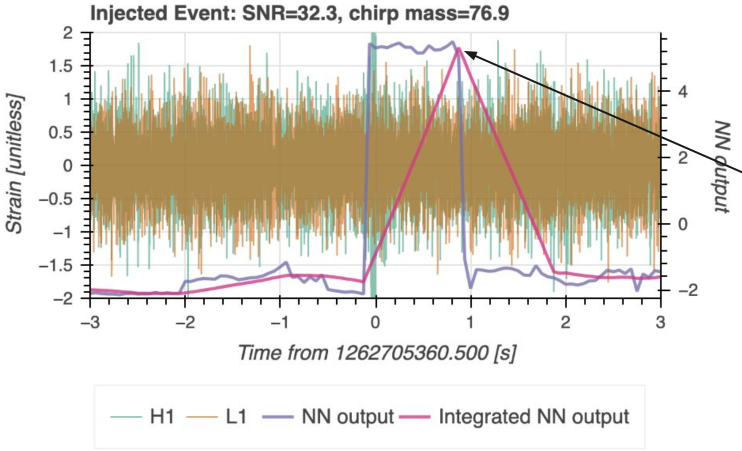
False alarm rates of these signals estimated by analyzing many background timeslides

Leverage triton inference server and hermes library to accelerate inference



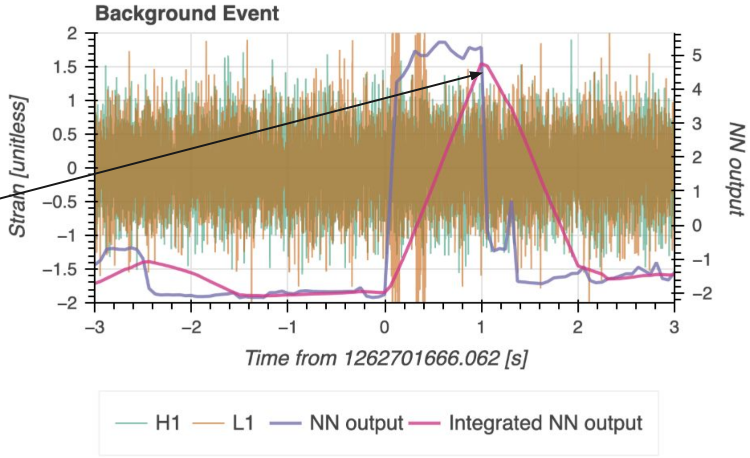
Processing Network Outputs

Visualization



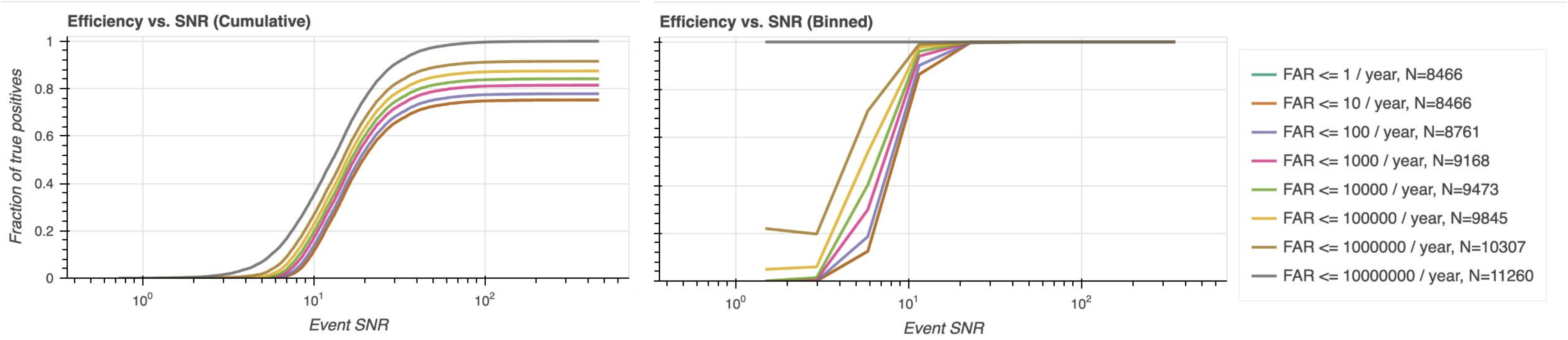
Example network output from simulated signal

Events



Example network output from timeshifted background events

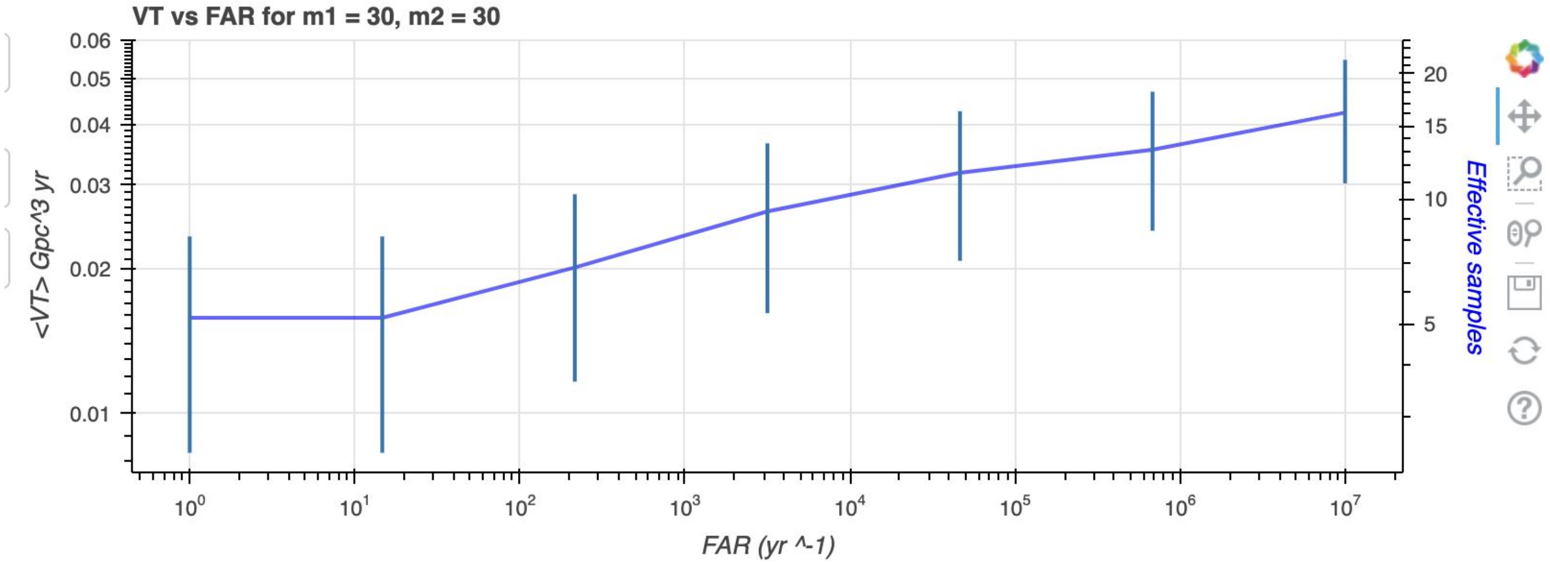
Performance Metrics



Efficiency vs. SNR (Cumulative)

Efficiency vs. SNR (Binned)

Performance Metrics

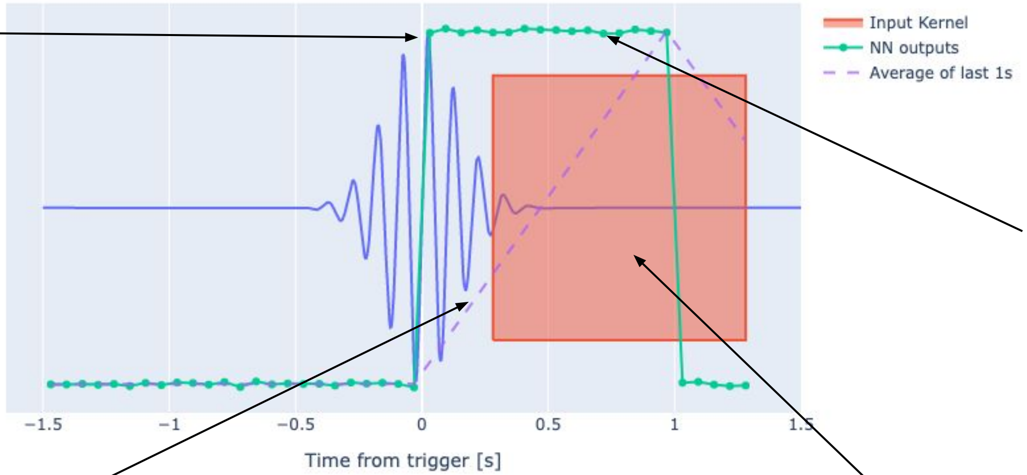


Next Steps

- Scaling up
  - Training Dataset: glitches, signals
  - Timeslide data analyzed → More significant detections
  - Larger Models
- Retraining schemes
  - How do we know when our model is stale?
  - How often do we have to retrain?

Backup Slides

Analyzing Network Outputs



As coalescence time enters window, network begins to ring

Green dots show raw network outputs

Dotted purple line shows average of last kernel length second of network outputs

Red sliding kernel represents current window of data evaluated by network