Draft version November 24, 2025
Typeset using LaTeX **twocolumn** style in AASTeX631

GSpyNetTreeS : a machine learning solution for glitch localization in time and frequency

Man Leong Chan,[1] Jess McIver,[1] Yannick Lecoeuche,[1] Dhatri Raghunathan,[2] Sofía Álvarez-López,[3] Julian Ding,[1] Annudesh Liyanage,[1] Raymond Ng,[4] and Heather Fong[1]

[1] *Department of Physics and Astronomy, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada*
[2] *Indian Institute of Technology Kharagpur, West Bengal, India*
[3] *Department of Physics and Kavli Institute for Astrophysics and Space Research, Massachusetts Institute of Technology, Cambridge, 02139, USA*
[4] *Department of Computer Science, University of British Columbia, Vancouver, British Columbia, V6T1Z4, Canada*

ABSTRACT

Data from ground-based gravitational wave detectors are often contaminated by non-Gaussian instrumental artifacts or detector noise transients. Unbiased source property estimation relies on the ability to correctly identify and characterize these artifacts and remove them if necessary. To this end, the LIGO-Virgo-KAGRA Collaboration has implemented candidate vetting for all significant candidates to identify the presence of artifacts and assess the need for mitigation. The current candidate vetting process requires human experts to identify the frequency ranges and the time windows associated with any data quality issues present. Differences in judgment between human experts may cause inconsistency, making results difficult to reproduce across gravitational wave events. We present `GSpyNetTreeS`, an extension to `GSpyNetTree` based on the You Only Look Once algorithm, for the automatic detection, classification, and time-frequency localization of detector noise transients. As a proof of concept, we tested `GSpyNetTreeS`'s performance on the data collected by the LIGO detectors during the third observing run for gravitational waves as well as common detector glitch classes included in `GSpyNetTree`: Blip, Low frequency blip, Low frequency line and Scratchy. We also demonstrated that `GSpyNetTreeS` is capable of accurately identifying common glitch classes and capturing the frequency and time information associated with detected detector noise transients, establishing its potential as an automatic event validation tool for LIGO-Virgo-KAGRA's observing runs.

1. INTRODUCTION

Since the first observation of gravitational waves (GWs) from a binary black hole merger (BBH) by the LIGO detectors (Aasi et al. 2015) in 2015 (Abbott et al. 2016), detection of GWs from compact binary coalescences (CBCs) has become commonplace. The LIGO-Virgo-KAGRA Collaboration (LVK) has reported observations of approximately 90 GWs from the first three LIGO-Virgo-KAGRA observing runs (Abbott et al. 2019, 2021a, 2023). With over 200 significant GW candidates already identified in the current observing run, fourth observing run (O4)[1], the catalog of GWs is expected to quickly expand. Thanks to improved LIGO detector sensitivity, there is also the potential to observe GWs from sources that have so far evaded detection such as core-collapse supernovae and spinning neutron stars (Abbott et al. 2022a, 2021b).

Ground-based GW detectors such as the LIGO, Virgo (Acernese et al. 2015) and KAGRA (Aso et al. 2013) detectors are prone to a high rate of non-Gaussian and non-stationary terrestrial noises, known as glitches (Soni et al. 2025b; Davis et al. 2021a; Abbott et al. 2020; Berger 2018). For example, the LIGO and Virgo detectors in the third observing run (O3) experienced a median glitch rate of approximately above 1 per minute, indicating a non-negligible probability ($\mathcal{O}(10\%)$) that a glitch overlaps a signal from CBCs (Davis et al. 2022). Depending on the morphology, the presence of a glitch can bias GW source property estimates (Macas et al. 2022; Payne et al. 2022; Ghonge et al. 2024; Hourihane et al. 2022; Pankow et al. 2018; Canton et al. 2014; Powell 2018; Udall et al. 2025) and even mimic GW signals, leading to false detection of GWs even after data conditioning and event rank-

[1] https://gracedb.ligo.org/superevents/public/O4/

ing statistics are applied, e.g. by CBC search pipelines GstLAL (Tsukada et al. 2023; Ewing et al. 2024), PyCBC (Dal Canton et al. 2021; Usman et al. 2016), MBTA (Alléné et al. 2025), and SPIIR (Chu et al. 2022).

To determine whether a GW candidate is affected by data quality issues, the LVK Collaboration has implemented candidate vetting (Soni et al. 2025b). Candidate vetting is a process where human experts cross check a list of data quality tasks[2] for a given GW candidate, to determine if data quality issues such as glitches are present and could affect source property estimation for the GW candidate. For example, tasks producing results for each GW candidate via the LIGO-Virgo Data Quality Report[3] in O4 include `iDQ`, a supervised learning tool that detects detector noise artifacts through auxiliary channel data (Biswas et al. 2013; Essick et al. 2013), `GSpyNetTree` (Alvarez-Lopez et al. 2024; Jarov et al. 2023), a machine learning classifier for the identification of different glitch morphologies, and `GlitchFind` (Vazsonyi & Davis 2023), a method to identify excess energy in the data surrounding a signal assuming a Gaussian noise background, etc. If a glitch is present, a human expert will manually identify the time and frequency region of the glitch for downstream data analyses such as glitch removal or noise subtraction (Davis et al. 2022), e.g. via the BayesWave algorithm (Cornish & Littenberg 2015). Accurate identification of glitches and their time-frequency region is essential to ensure reliable estimates of GW source properties.

However, human validation of a candidate, including the identification of the time-frequency region of any glitches present is time intensive and can introduce inconsistencies, making validation results difficult to reproduce. An expected increase in event rate for future observing runs that will leverage more sensitive GW detectors[4] will significantly exacerbate this challenge.

In recent years, machine learning techniques and their applications have shown promising potential in many scientific fields including GW astronomy. In particular, a machine learning technique for image segmentation known as You Only Look Once (YOLO) (Redmon et al. 2016), a convolutional neural network based technique, has been gaining popularity due to its efficiency and accuracy in locating objects of interest in the form of pixel coordinates from input images (Wang et al. 2024).

In this work, we present as a proof of concept a novel algorithm based on YOLO for the automatic identification, classification, and localization of glitches in time and frequency. The algorithm, referred to as `GSpyNetTreeS`, is an extension to the `GSpyNetTree` algorithm. In addition to classifying multiple glitches in GW strain data simultaneously, `GSpyNetTreeS` is capable of estimating the frequency ranges and time windows during which the glitches occur. The estimates of the frequency and time ranges can then be used directly as inputs for downstream tasks such as noise subtraction. Incorporating `GSpyNetTreeS` in candidate vetting will enable fast, automatic and fully reproducible identification of the presence of data quality issues and reduce the need for human expert input, contributing to a more robust work flow.

This manuscript is organized as follows: in Section 2, we will describe the development of `GSpyNetTreeS` including the generation of samples for training and testing; in Section 3, we will present the performance of `GSpyNetTreeS` on the testing samples; in Section 4 we conclude with a discussion of potential future applications.

2. GSpyNetTreeS

Deep learning algorithms such as convolutional neural networks have had numerous prior applications to GW astronomy, including real-time validation of events (Cabero et al. 2020; Chan et al. 2024; Abbott et al. 2022b; Raza et al. 2024), the identification of detector glitches (Alvarez-Lopez et al. 2024; Zevin et al. 2017, 2024; Glanzer et al. 2021; Wu et al. 2025; Koyama et al. 2024), the detection of astrophysical signals (Gabbard et al. 2018; Chan et al. 2020; Skliris et al. 2024; Mishra et al. 2022), GW source property estimation (Gabbard et al. 2022; Langendorff et al. 2023; Green et al. 2020), sky localization (Chatterjee et al. 2023), signal denoising (Murali & Lumley 2023), and detector noise transient generation (Powell et al. 2023) (for a review, see Cuoco et al. (2021)). Compared to more traditional methods for GW data analysis, an advantage of deep learning algorithms is that they are relatively computationally cheap to run while the performance is comparable (Heaton 2018).

`GSpyNetTreeS` leverages the YOLO family, a class of deep learning algorithms designed for object detection in images with convolutional neural networks. Object detection in this context means the identification, the localization and the classification of objects of interest in a given image. YOLO addresses these issues as a single regression problem, predicting both the locations and class probabilities of objects in input images. This approach allows for fast, efficient and accurate detection, making it suitable for real-time applications.

[2] https://detchar.docs.ligo.org/dqrtasks/index.html
[3] https://docs.ligo.org/detchar/data-quality-report/
[4] https://observing.docs.ligo.org/plan/

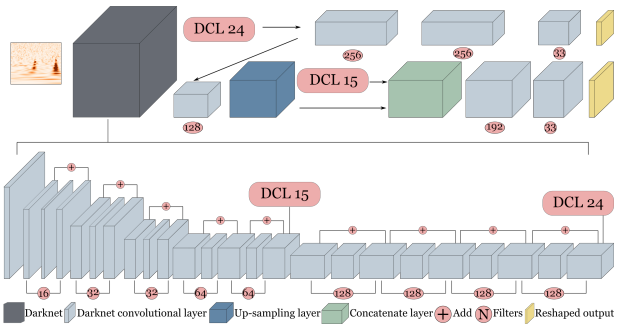
Many versions of YOLO have been developed in the literature, from YOLO to YOLOv11 (Khanam & Hussain 2024) and beyond (Cheng et al. 2024). Each version incorporates innovations to address different issues and challenges (see Terven et al. (2023); Jiang et al. (2022); Alif & Hussain (2024); Wang et al. (2024) for reviews and (Soni et al. 2025a) for a complementary application of YOLO for GW detection). In this work, we adopt an architecture that is closest to YOLOv3 due to its balance of speed, accuracy and ease of implementation (Redmon & Farhadi 2018) while specific changes are made to allow `GSpyNetTreeS` to detect, locate and classify glitches and GWs given input data.

As shown in Figure 1, `GSpyNetTreeS`'s architecture includes a Darknet component for processing time-frequency representation of GW detector data (see Section 2.1). The outputs from the Darknet component are then passed on to subsequent layers for further processing. The outputs of `GSpyNetTreeS` are two multi-dimensional arrays, containing information on the frequency boundaries, time windows and classification for glitches and GWs detected in input images, allowing `GSpyNetTreeS` to automatically provide useful information for event validation and downstream data analysis tasks such as glitch removal or noise subtraction. An example is shown in Figure 2.

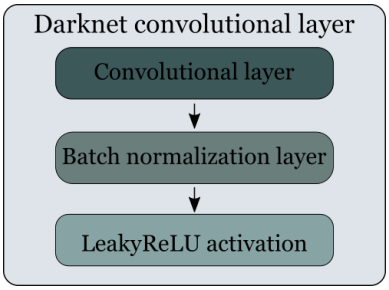
2.1. *Data preparation*

To train `GSpyNetTreeS` for the identification, localization and classification of glitches and GWs, we use the `GSpyNetTree` data set described in Alvarez-Lopez et al. (2024). This data set consists of glitches observed in the LIGO Hanford and LIGO Livingston detectors during O3 (Abbott et al. 2021a, 2023), obtained from the Gravity Spy classifications (Glanzer et al. 2021) via LIGO-DV web (Areeda et al. 2017). As a proof of concept we include in our data sets only Blips, Low frequency blips, Low frequency lines, Scattering and Scratchy[5] as these are some of the most commonly occurring glitch classes (Davis et al. 2021b; Soni et al. 2025b). The GW signals were simulated assuming the waveform model IMRPhenomPv2 (Khan et al. 2016; Husa et al. 2016) and added to segments of 64-second quiet detector data from both the LIGO Hanford and LIGO Livingston detectors during previous observing runs, using the inspiral injection module of LALSuite (LIGO Scientific Collaboration 2018). For this proof-of-principle study, to compare performance against commonly occurring glitches we focus on higher mass BBH sources which are of $\lesssim$ 1-second duration in the detectors' sensitive frequency range. GW examples were uniformly drawn from a total merger mass range of 5M to 350M, with individual masses ranging from $2M_\odot$ to $175M_\odot$, an SNR range of 8 to 35, and individual component spins ranging from 0.05 to 0.95.

[5] Note that LIGO detector glitch names are often derived from their time-frequency morphology (e.g. Blips, Lines) or source (e.g. Scattering)



(a)



(b)

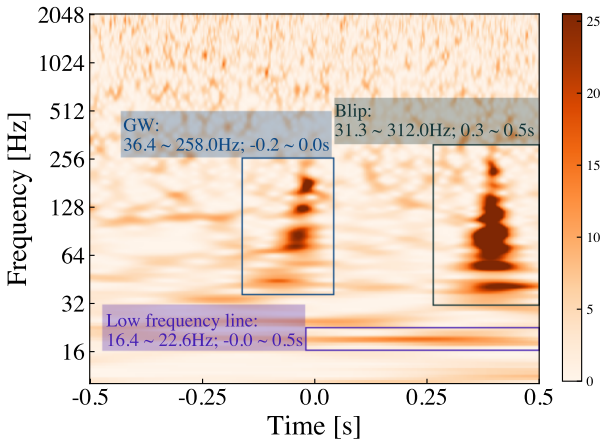
Figure 1. The architecture of `GSpyNetTreeS`. The input to `GSpyNetTreeS` is a time-frequency representation image of GW data. `GSpyNetTreeS` consists of two major components. The first component is a simplified Darknet with 24 Darknet convolutional layers. The Darknet generates two sets of outputs from its 15th and 24th layer. The outputs will be further processed by the second component of `GSpyNetTreeS`, which consists of a few Darknet convolutional layers, an up-sampling layer and a concatenate layer. The outputs are then passed onto two output layers, each of which is specialized for regions of excess power of different sizes (see Section 2.1). The lower panel shows the layers used to construct a Darknet convolutional layer (DCL).

Time-frequency representations of the data set are then generated using the Q-transform (Chatterji et al. 2004), a time-frequency visualization tool commonly used for characterizing GW strain data. The Q-transform is a modification of the short-time Fourier transform that tiles the time-frequency plane with pixels of constant quality factor Q. This is achieved by using analysis windows with durations inversely proportional to the frequency, resulting in different resolutions at different frequencies. In this proof-of-concept

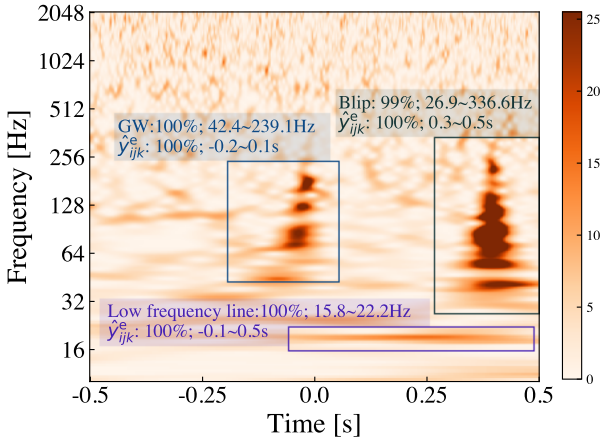
work, we use a time-frequency representation of 1 second in duration with a quality factor Q of 25, consistent the feature sets of image classification algorithms `Gravity Spy` (Zevin et al. 2017, 2024; Wu et al. 2025) and `GSpyNetTree` (Alvarez-Lopez et al. 2024), which leverage similar images of different durations. The time-frequency representations are converted to images of 280 pixels by 340 pixels, which are then normalized such that the values of the pixels range from 0 to 25.5, with higher values indicating larger energy as illustrated in Figure 2.

In order to train `GSpyNetTreeS`, we first need to define the time-frequency region corresponding to glitches and GWs. We considered glitches and GWs loosely as regions of excess power of different morphologies present in time-frequency representation of GW detector data. We defined a region of excess power in the time-frequency representation as a cluster of image pixels within which the number of connected neighboring loud pixels is ≥ 28. Loud pixels are defined as image pixels where the normalized pixel energy is ≥ 10.5. This value is chosen such that loud pixels are image pixels with energy greater than 98% of all pixels in the data set as shown in Figure 3. Based on these criteria, however, some simulated GW signals were too weak and are thus not recognized as regions of excess power. These signals are therefore excluded from our data. Excluding these signals will undermine `GSpyNetTreeS`'s capability to detect weak GWs, however, this is not a limitation to `GSpyNetTreeS`'s intended purpose for the identification of data quality issues for event validation. In addition, these threshold choices can be easily adjusted prior to training to incorporate weaker examples, if desired.

The time-frequency region containing each instance of excess power identified by the above criteria was then labeled manually as a bounding box, with human expert judgment used to assess the most appropriate time-frequency bounds. The class (e.g. glitch type) was labeled according to the `GSpyNetTree` data set classification. Figure 2.1 shows an example generated from O3 LIGO Hanford data, with three manually identified and labeled regions of excess power. In total, 13 000 unique time-frequency representation images were generated. The time-frequency representation images were then augmented by applying a random time shift of at most 0.25 s to the GW strain data in time prior to generation of time-frequency representation of the data, resulting in 23 749 time-frequency representation images in total. This augmentation improves the algorithm's robustness to changes in noise background, as demonstrated in (Alvarez-Lopez et al. 2024). The number of regions of excess power identified for each class (GWs and glitches) is given in Table 1. Approximately 10% of



(a)



(b)

Figure 2. Images of time-frequency representation of a time series from LIGO Hanford. The images show three regions of excess power that meet the criteria defined in Section 2.1. A simulated GW signal from a binary black hole merger with component masses 35.6 and 40.3 $M_\odot$ is shown in the blue time-frequency box. The other two time-frequency boxes highlight real glitches observed by the detector: a Blip (green box) and Low frequency line (red box). The upper panel shows the 'ground truth': manually drawn bounding boxes, with human expert judgment used to assess the most appropriate time-frequency bounds. The lower panel shows the corresponding bounding boxes predicted by the algorithm, including predicted frequency ranges and time windows. $\hat{y}^e_{ijk}$ indicates the confidence of `GSpyNetTreeS` that the predicted bounding boxes surround a region of excess power (see Section 2.2).

the data were randomly selected and reserved for testing, and an additional 10% were reserved for validation during training.

Image segmentation algorithms require clear coaching on their target output, which in our case is time-

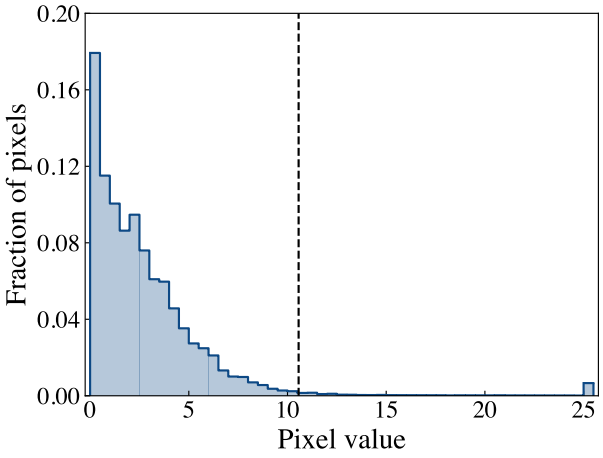


Figure 3. A histogram showing the distribution of pixel values for all images of time-frequency representation of GW detector data in the training samples. The vertical line indicates the threshold for loud pixels (i.e., 10.5), which is greater than 98% of all pixels.

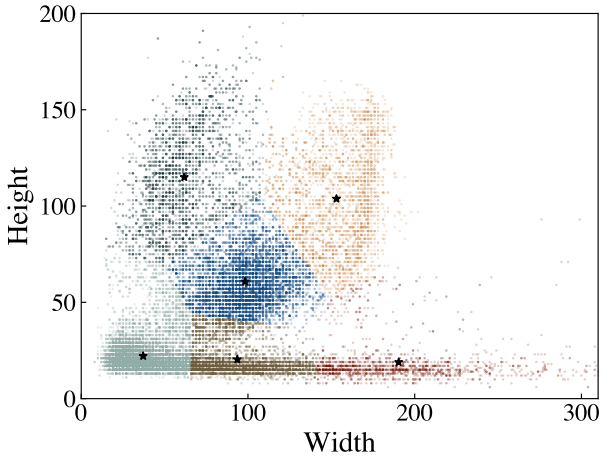


Figure 4. A scatter plot showing the distribution of the widths and heights in pixels for all the bounding boxes in the images for training. K-means clustering is applied to group bounding boxes into different clusters, with cluster members shown in different colours. The centroids of the clusters are then taken as the *typical size* of anchor boxes, which are used as reference boxes for `GSpyNetTreeS`.

Morphology	Training	Validation	Testing
GW	11479	1249	1385
Blip	3993	464	513
Low frequency blip	3440	374	449
Low frequency line	5767	651	667
Scattering	1832	198	250
Scratchy	19382	2124	2419
Images	19236	2138	2375

Table 1. The number of regions of excess power identified for each class based on morphology. The last row indicates the number of images for training, validation and test To train `GSpyNetTreeS`, approximately 10% of the total samples are randomly drawn separately for validation and testing.

frequency bounding boxes. In order to help simplify `GSpyNetTreeS`'s task we fed it the typical sizes of time-frequency boxes by applying K-means clustering to bounding boxes' widths and heights from the images set aside for training. We identified six distinct groups of time-freqeuncy bounding boxes, as illustrated in Figure 4. The centroids of these bounding box clusters form *anchor boxes*, which act as reference bounding boxes for each cluster.

For each time-frequency representation image, six grids covering the entire image are then generated, with each associated with one of the six anchor boxes. The bounding box for each manually identified region of excess power is then matched to a grid cell from a grid associated with the anchor box that most closely matches the bounding box's dimensions, as determined using K-means clustering. The specific grid cell responsible for the region of excess power is determined by locating the grid cell containing the center of that region. Each grid cell can be associated with one region of excess power at most.

The assigned grid cell is then labeled with the bounding box's normalized width, height, and center coordinates relative to the cell, as well as the predicted morphological class of the region. The use of anchor boxes allows each grid to specialize in detecting objects of a specific size range, thus improving the model's performance in identifying differing regions of excess power in time-frequency images. To further optimize the performance of `GSpyNetTreeS` for regions of excess power of different sizes, these grids are divided into two groups with different number of grid cells. The bounding box information in each group of the grids is then the target output for one of the output layers as shown in Figure 1. The number of grid cells for each grid and the size of the associated anchor boxes are given in Table 2.

An example prediction from `GSpyNetTreeS` is given in Figure 2.1.

2.2. *Loss function*

Machine learning algorithms are generally trained by minimizing their loss functions. Loss functions measure and quantify the error between a machine learning algorithm's outputs given an input and the ground truth. During training, small subsets of training data are passed through the algorithm in multiple iterations. The back-propagation algorithm then calculates the gra-

Output	Grid	Grid cells	Anchor box
	1		(61.67, 114.97)
1	2	(17, 21)	(152.95, 103.72)
	3		(190.53, 18.91)
	4		(37.17, 22.15)
2	5	(35, 42)	(93.65, 20.28)
	6		(98.22, 60.72)

Table 2. The number of grids, grid cells and the sizes of the associated anchor boxes. The first column indicates the outputs from `GSpyNetTreeS` (see Figure 1). Each output from `GSpyNetTreeS` contains predicted bounding box information for three grids. The second column indicates the grid and the third column the number of grid cells, i.e., (row, column) for each of the grids. The sizes of the associated anchor boxes in pixel are shown in the fourth column.

dient of the loss function with respect to the trainable parameters of the algorithm, such as the weights and biases of the neurons in each layer. These parameters are then updated using a gradient descent algorithm in order to iteratively minimize the loss-function. Once this process converges and the loss function is minimized, the training is considered complete. The algorithm will be able to take in input in the form of new data and generate outputs that best correspond to the learned mappings between input and output during training. An appropriate loss function is therefore important to achieve the desired performance.

`GSpyNetTreeS` is designed to detect, locate and classify regions of excess power in a given time-frequency representation image representing GW detector data. This involves 1. identifying the existence of regions of excess power, 2. determining the positions of the corresponding bounding boxes, 3. estimating the width and height of the boxes, and 4. classifying the morphology of the detected regions of excess power. As such, the loss function must incorporate each of these objectives. For each input time-frequency representation image, we therefore employ the following loss function,

$$\begin{aligned} &L_{\text{Total}}(\mathbf{y}^{\text{e}}, \mathbf{y}^{\text{p}}, \mathbf{y}^{\text{wh}}, \mathbf{y}^{\text{C}}, \hat{\mathbf{y}}^{\text{e}}, \hat{\mathbf{y}}^{\text{p}}, \hat{\mathbf{y}}^{\text{wh}}, \hat{\mathbf{y}}^{\text{C}}) \\ &\quad = L_{\text{e}}(\mathbf{y}^{\text{e}}, \hat{\mathbf{y}}^{\text{e}}) + L_{\text{p}}(\mathbf{y}^{\text{p}}, \hat{\mathbf{y}}^{\text{p}}) + \\ &\quad\quad L_{\text{wh}}(\mathbf{y}^{\text{wh}}, \hat{\mathbf{y}}^{\text{wh}}) + L_{\text{C}}(\mathbf{y}^{\text{C}}, \hat{\mathbf{y}}^{\text{C}}), \end{aligned} \tag{1}$$

where L_{Total} is the total loss and L_{e}, L_{p}, L_{wh} and L_{C} are the losses for the absence or presence of any regions of excess power in the image, the positions of the corresponding bounding boxes, the widths and heights of the boxes and the classification of the regions of excess power respectively. The losses are defined below,

$$\begin{aligned} &L_{\text{e}}(\mathbf{y}^{\text{e}}, \hat{\mathbf{y}}^{\text{e}}) = \frac{1}{N_{\text{g}}} \sum_{k}^{N_{\text{g}}} \frac{1}{N_{\text{gc}}(k)} \times \\ &\quad \sum_{i,j}^{N_{\text{gc}}(k)} \{m_{ijk}[y^{\text{e}}_{ijk} \log \hat{y}^{\text{e}}_{ijk} + (1 - y^{\text{e}}_{ijk}) \log(1 - \hat{y}^{\text{e}}_{ijk})] + \\ &(1 - m_{ijk}) n_{ijk}[y^{\text{e}}_{ijk} \log \hat{y}^{\text{e}}_{ijk} + (1 - y^{\text{e}}_{ijk}) \log(1 - \hat{y}^{\text{e}}_{ijk})]\}; \\ &L_{\text{p}}(\mathbf{y}^{\text{p}}, \hat{\mathbf{y}}^{\text{p}}) = \frac{1}{N_{\text{g}}} \sum_{k}^{N_{\text{g}}} \frac{1}{N_{\text{gc}}(k)} \times \\ &\quad \sum_{i,j}^{N_{\text{gc}}(k)} m_{ijk}\{(y^{\text{px}}_{ijk} - \hat{y}^{\text{px}}_{ijk})^2 + (y^{\text{py}}_{ijk} - \hat{y}^{\text{py}}_{ijk})^2\}; \\ &L_{\text{wh}}(\mathbf{y}^{\text{wh}}, \hat{\mathbf{y}}^{\text{wh}}) = \frac{1}{N_{\text{g}}} \sum_{k}^{N_{\text{g}}} \frac{1}{N_{\text{gc}}(k)} \times \\ &\quad \sum_{i,j}^{N_{\text{gc}}(k)} m_{ijk}\{(y^{\text{w}}_{ijk} - \hat{y}^{\text{w}}_{ijk})^2 + (y^{\text{h}}_{ijk} - \hat{y}^{\text{h}}_{ijk})^2\}; \\ &L_{\text{C}}(\mathbf{y}^{\text{C}}, \hat{\mathbf{y}}^{\text{C}}) = \frac{1}{N_{\text{g}}} \sum_{k}^{N_{\text{g}}} \frac{1}{N_{\text{gc}}(k)} \times \\ &\quad \sum_{i,j}^{N_{\text{gc}}(k)} m_{ijk} \sum_{c}^{N_{\text{class}}} y^{\text{C}}_{ijkc} \log(\hat{y}^{\text{C}}_{ijkc}). \end{aligned} \tag{2}$$

In the above equations, $\mathbf{y}^{\text{e}}$ is a three dimensional array representing the ground truth for the presence or absence of regions of excess power in the entire image. Its element y^{e}_{ijk} represents the ground truth for the presence or absence of any region of excess power associated with the grid cell at the i th row and j th column of the k th grid. The value of y^{e}_{ijk} can be either 1 (presence) or 0 (absence). $\mathbf{y}^{\text{p}}$ is an array composed of two separate three-dimensional arrays, $\mathbf{y}^{\text{px}}$ and $\mathbf{y}^{\text{py}}$ representing the ground truth coordinates of the center of the bounding boxes corresponding to the regions of excess power. For a bounding box of which the center falls within the grid cell at the i th row and j th column of the k th grid, both the horizontal coordinate y^{px}_{ijk} and vertical coordinate y^{py}_{ijk} are linearly continuous and are determined by the relative position of the center to the grid cell. A coordinate of (1, 1) for (y^{px}_{ijk}, y^{py}_{ijk}) indicates that the center is at the bottom rightmost of the grid cell while (0, 0) at the top leftmost. The widths and heights of all the bounding boxes are expressed as $\mathbf{y}^{\text{wh}}$, which is similarly composed of two separate three-dimensional arrays $\mathbf{y}^{\text{w}}$ and $\mathbf{y}^{\text{h}}$ indicating the widths and heights respectively. The width y^{w}_{ijk} and the height y^{h}_{ijk} of a bounding box associated with the grid cell at the i th row and j th column of the k th grid are expressed in relation to the

anchor box associated with the grid, as given below

$$\begin{aligned} \mathrm{B_w} &= \mathrm{A_{w}}_k \exp(y^{\mathrm{w}}_{ijk}); \\ \mathrm{B_h} &= \mathrm{A_{h}}_k \exp(y^{\mathrm{h}}_{ijk}), \end{aligned} \tag{3}$$

where $\mathrm{B_w}$, $\mathrm{B_h}$, $\mathrm{A_w}_k$ and $\mathrm{A_h}_k$ are the widths and heights for the bounding box and the associated anchor box respectively.

Finally, $\mathbf{y}^{\mathrm{C}}$ is a multi-dimensional array. Its element $\mathbf{y}^{\mathrm{C}}_{ijk}$ is a one-dimensional array where the number of elements is equal to the number of considered classes, corresponding to the ground truth classification of a region of excess power whose center falls within the grid cell at the i th row and j th column of the k th grid. The corresponding outputs from `GSpyNetTreeS` for the grid cell at the i th row and j th column of the k th grid are $\hat{y}^{\mathrm{e}}_{ijk}$, $\hat{y}^{\mathrm{px}}_{ijk}$, $\hat{y}^{\mathrm{py}}_{ijk}$, $\hat{y}^{\mathrm{w}}_{ijk}$, $\hat{y}^{\mathrm{h}}_{ijk}$ and $\hat{\mathbf{y}}^{\mathrm{C}}_{ijk}$. Unlike its ground truth counterpart y^{e}_{ijk}, however, $\hat{y}^{\mathrm{e}}_{ijk}$ is a value ranging from 0 to 1 indicating the `GSpyNetTreeS`'s confidence that a region of excess power is present and associated with the grid cell at the i th row and j th column of the k th grid. In addition, similar to $\mathbf{y}^{\mathrm{C}}_{ijk}$, $\hat{\mathbf{y}}^{\mathrm{C}}_{ijk}$ is a one-dimensional array where each element ranges from 0 to 1 corresponding to `GSpyNetTreeS`'s confidence that the detected region of excess power belongs to each of the classes. The elements of $\hat{\mathbf{y}}^{\mathrm{C}}_{ijk}$ sum to 1. In the equations, the summations sum over the number of grids N_{g}, the number of grid cells in the k th grid $N_{\mathrm{gc}}(k)$, and the number of classes N_{class}. To allow for an optimal performance, the losses are modeled differently. In particular, we employ binary cross-entropy for L_{e}, mean squared errors for both L_{p} and L_{wh}, and categorical cross-entropy for L_{c}.

For any given image, the number of grid cells that are not associated with any regions of excess power in the image will likely vastly outnumber regions of excess power. As a result, the loss function may disproportionately emphasize these empty grid cells, driving `GSpyNetTreeS` to overwhelmingly predict the absence of region of excess power. To account for this, a mask is applied to the loss function. If the grid cell at the i th row and j th column of the k th grid is associated with a region of excess power, the value of m_{ijk} will be 1 and 0 otherwise. Applying the mask removes the effects of empty cells and allows the model to focus on grids cells associated with regions of excess power. Unfortunately, for L_{e}, this may have the negative effect of achieving an algorithm that will always generate $\hat{y}^{\mathrm{e}}_{ijk}$ close to 1, over predicting the existence of regions of excess power. This is because for empty grid cells (i.e., m_{ijk} equals 0), multiplying m_{ijk} prevents the loss function from penalizing the algorithm for generating non-zero $\hat{y}^{\mathrm{e}}_{ijk}$. To counter this effect, we apply a term n_{ijk} to the calculation of L_{e}. The value of n_{ijk} is determined by computing the value of Intersection over Union (IoU) (see Section 3) between the predicted bounding boxes at the i th row and j th column of the k th grid and each ground truth bounding box in the image. If the IoU for at least one ground truth bounding box and the predicted bounding box is greater than 0.3, n_{ijk} is 1 and 0 otherwise. The losses are then averaged over the number of grid cells $N_{\mathrm{gc}}(k)$ in the k th grid and the number of grids N_{g}. The training of `GSpyNetTreeS` is complete when the losses in Eq. 2 are minimized.

3. PERFORMANCE

To show the performance of `GSpyNetTreeS` on the testing samples given in Table 1, it is necessary to introduce the concept of IoU (Rezatofighi et al. 2019). IoU is a metric used to measure and evaluate the match between two boxes with given sizes and positions. For two boxes B^1 and B^2, IoU is given by

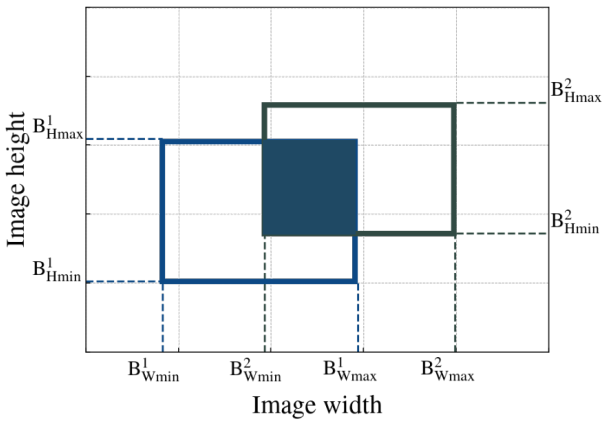
$$\mathrm{IoU} = \frac{\mathrm{A_{overlap}}}{\mathrm{A_{union}}}, \tag{4}$$

where $\mathrm{A_{overlap}}$ and $\mathrm{A_{union}}$ are the overlapping area and the combined area of the boxes, as illustrated in Figure 5 and given by

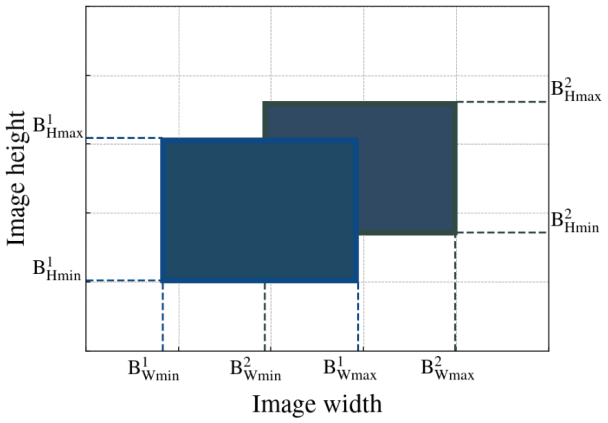
$$\begin{aligned} \mathrm{A_{overlap}} &= (\mathrm{B}^1_{\mathrm{W_{max}}} - \mathrm{B}^2_{\mathrm{W_{min}}})(\mathrm{B}^1_{\mathrm{H_{max}}} - \mathrm{B}^2_{\mathrm{H_{min}}}) \\ \mathrm{A_{union}} &= (\mathrm{B}^1_{\mathrm{W_{max}}} - \mathrm{B}^1_{\mathrm{W_{min}}})(\mathrm{B}^1_{\mathrm{H_{max}}} - \mathrm{B}^1_{\mathrm{H_{min}}}) + \\ &\quad (\mathrm{B}^2_{\mathrm{W_{max}}} - \mathrm{B}^2_{\mathrm{W_{min}}})(\mathrm{B}^2_{\mathrm{H_{max}}} - \mathrm{B}^2_{\mathrm{H_{min}}}) - \\ &\quad \mathrm{A_{overlap}}, \end{aligned} \tag{5}$$

where $\mathrm{B}^{1/2}_{\mathrm{W_{max/min}}}$ and $\mathrm{B}^{1/2}_{\mathrm{H_{max/min}}}$ are the coordinates of the boxes as shown in Figure 5. For any given pair of boxes, IoU ranges from 0 to 1, with 0 indicating that the boxes do not have any overlap and 1 indicating that the boxes completely overlap with each other.

For an input image, multiple nearby grid cells in the outputs from `GSpyNetTreeS` may pick up the same region of excess power, generating different bounding boxes, essentially containing the same information, albeit with varying degree of confidence for the presence of the region of excess power (i.e., $\hat{y}^{\mathrm{e}}_{ijk}$). To select the bounding boxes that represent the algorithm's most confident detections, a threshold $\mathrm{T_e}$ on detection confidence (i.e., $\hat{y}^{\mathrm{e}}_{ijk}$) is selected. The values of $\hat{y}^{\mathrm{e}}_{ijk}$ for all grid cells passing $\mathrm{T_e}$ are ranked and the bounding box with the maximum confidence is then marked as the first valid detection. The values of IoU between the selected bounding box and each of the remaining bounding boxes will then be computed. If for a bounding box, the value of IoU is larger than a pre-selected threshold $\mathrm{T_{IoU}}$, the bounding box will be considered repeated detection and



(a)



(b)

Figure 5. The definition of IoU. Given two boxes B^1 and B^2, the shaded areas in the top and bottom panels show the overlapping area between the two boxes $A_{overlap}$ and the combined area of the boxes A_{union} respectively.

be discarded. After all the bounding boxes with IoU $>$ T_{IoU} are discarded, the remaining bounding boxes will have their $\hat{y}^e_{ijk}$ ranked again and the bounding boxes with the maximum $\hat{y}^e_{ijk}$ will be marked as the second valid detection. This process will repeat until no IoU between any pair of bounding boxes is larger than T_{IoU} and the maximum $\hat{y}^e_{ijk}$ for the remaining bounding boxes is $< T_e$. The selected set of bounding boxes then represents the algorithm's confident detections of regions of excess power in the input image. In the remainder of this paper, we use a T_e of 0.5 and a T_{IoU} of 0.3.

We will first show the performance of `GSpyNetTreeS` in detecting the presence of regions of excess power as well as the classifications of the detected regions of excess power. Since for `GSpyNetTreeS`, a correct identification of a region of excess power requires the correct classification and estimation of the position and size of

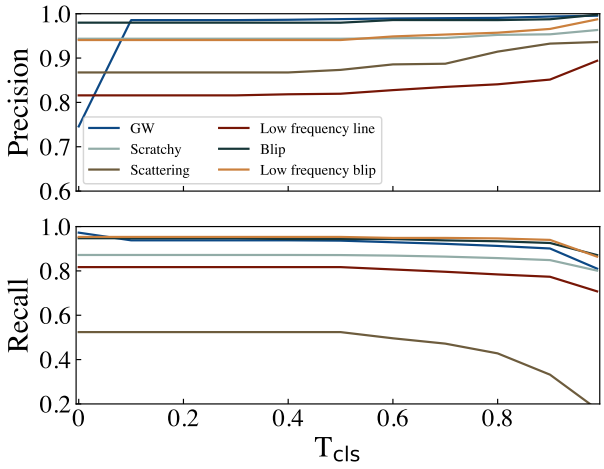


Figure 6. The performance of `GSpyNetTreeS` shown as curves of precision (upper panel) and recall (lower panel). The curves are produced using a T_e of 0.5 and a T_{IoU} of 0.3 respectively.

the bounding box, we define true positives as valid detections with the following requirements: 1. the IoU between the predicted bounding box and the ground truth bounding box is $\geq T_{IoU}$, and 2. the morphology is correctly classified based on a threshold T_{cls}. False positives are defined as valid detections with the following conditions: 1. the predicted bounding box either does not have an associated ground truth bounding box where the IoU is $\geq T_{IoU}$, or 2. the classification of the morphology is incorrect based on T_{cls}. False negatives are defined as a ground truth region of excess power with the following conditions: 1. no corresponding predicted bounding box with an IoU $\geq T_{IoU}$, or 2. the classification of the morphology is incorrect based on T_{cls} for all predicted bounding boxes with an IoU $\geq T_{IoU}$. However, it should be noted that true negative does not have a straightforward definition. For this reason, we use curves of precision and recall as a function of T_{cls} in Figure 6 to quantify the performance instead of the more used receiver operator characteristic curve, which shows the true positive rate of a classifier as a function of false positive rate. For a given T_{cls}, the precision and recall are given below

$$\begin{aligned} \text{Precision} &= \frac{N_{TP}}{N_{TP}+N_{FP}}; \\ \text{Recall} &= \frac{N_{TP}}{N_{TP}+N_{FN}}, \end{aligned} \tag{6}$$

where N_{TP}, N_{FP} and N_{FN} are the number of true positives, false positives and false negatives. Precision indicates the proportion of correctly identified true positives out of all valid detections made by `GSpyNetTreeS`, reflecting its tendency to avoid false positive. Recall mea-

Morphology	N_{TP}	N_{FP}	N_{FN}
GW	1297(93.7%)	16(0.1%)	88(6.3%)
Blip	484(94.3%)	10(0.2%)	29(5.7%)
Low frequency blip	428(95.3%)	27(6.0%)	21(4.7%)
Low frequency line	545(82.0%)	120(18.0%)	122(18.0%)
Scattering	131(52.4%)	19(7.6%)	119(47.6%)
Scratchy	2108(87.2%)	126(5.2%)	311(12.8%)

Table 3. The performance of `GSpyNetTreeS` quantified by the true positives N_{TP}, false positives N_{FP} and false negatives N_{FN}. This table assumes a $\mathrm{T_{IoU}}$ of 0.3 and a $\mathrm{T_{cls}}$ of 0.5. The percentages provided in the brackets for N_{TP} and N_{FN} are true positive rates and false negative rates. As for `GSpyNetTreeS`, there is no straightforward definition of false positive rate, the percentages provided in the brackets for N_{FP} are N_{FP} divided by the total number testing samples (see Table 1).

sures the proportion of true positives out of all actual regions of excess power, reflecting the `GSpyNetTreeS`'s ability to identify relevant regions of excess power in the images. These metrics provide an overall accuracy of the algorithm in detecting and classifying the presence of regions of excess power.

As shown in Figure 6, both the precision and recall curves change only slightly over the range of $\mathrm{T_{cls}}$. This indicates that the performance of `GSpyNetTreeS` only depend on $\mathrm{T_{cls}}$ very weakly. This results from `GSpyNetTreeS`'s tendency to generate classification confidence $\hat{y}^{\mathrm{C}}_{ijkc}$ close to 0 or 1 and is therefore not affected by $\mathrm{T_{cls}}$ until $\mathrm{T_{cls}}$ becomes large. For reference, the N_{TP}, N_{FP} and N_{FN} at $\mathrm{T_{cls}}$ equal to 0.5 are provided in Table 3. The performance for GW, Blip and Low frequency blip is promising. For these three classes, the true positive rates are over 93% while the false negative rates are low single digits. However, for GW, this likely results in part from the training set exclusion of GW signals to weak to generate excess power meeting the criteria in Section 2. The performance for Scratchy is poorer despite this class having the largest number of samples. This is because using our definition of region of excess power provided in Section 2, many glitches classified as Scratchy are small regions with energy only slightly above the background. An example is shown in Figure 7. Some Scratchy glitches would be considered background if a different quality factor Q was used or if the time series was slightly shifted in time, altering the background estimation. To a lesser extent, this issue also affects Low frequency lines. The performance for Scattering is impacted by both this problem and insufficient training samples. We expect a more refined definition of the region of excess power (as well as a more refined

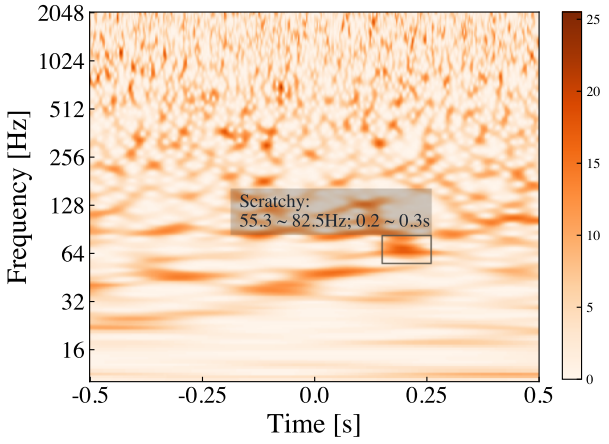
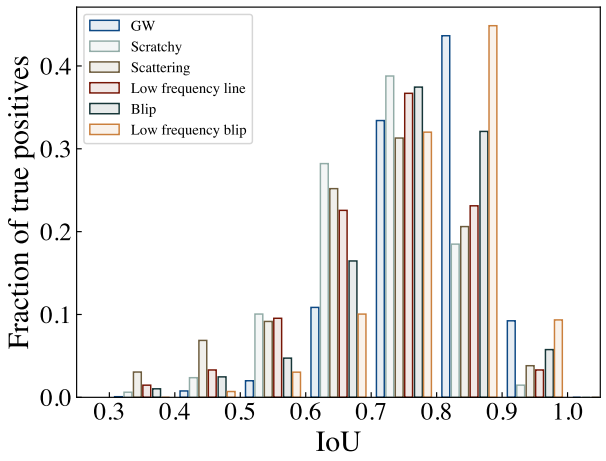


Figure 7. A time-frequency representation of a one-second segment of the data collected by LIGO Livingston during O3. A region of excess power is manually identified and classified as a Scratchy based on the definition provided in Section 2.1. The region of excess power is not detected by `GSpyNetTreeS` as it barely stands out from the background.

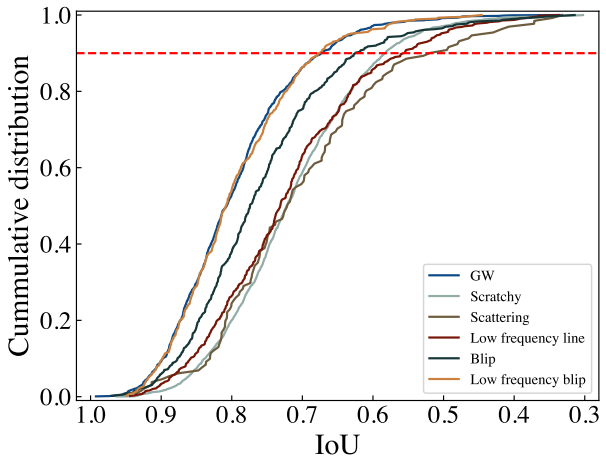
Q value) that better accounts for background variations will help improve performance.

Commonly used metrics for classifiers such as precision and recall are generally not correlated with how accurate the predicted bounding boxes are compared to the corresponding ground-truth bounding boxes. For example, two object detection algorithms with the same precision and recall can still perform vastly differently in terms of predicting bounding boxes accurately. To comprehensively assess `GSpyNetTreeS`'s performance it is necessary to understand whether the predicted bounding boxes (and associated frequency ranges and time windows of the regions of excess power) are consistent with the ground truth. For this purpose, we use IoU to quantify how well the predicted bounding boxes match the ground truths.

We show in Figure 3 the distribution of IoU for the valid detections that have a corresponding ground truth box (i.e., the true positives). The distributions for all glitches and GWs are cut off at 0.3 because of the selected value of $\mathrm{T_{IoU}}$. It can be seen that for most classes, the performance is similar with the peaks of the distributions centered between 0.7 - 0.9. Figure 3 shows the cumulative distributions of IoU. The dashed horizontal line indicates that 90% of correctly identified GW, Scratchy, Scattering, Low frequency line, Blip, and Low frequency blip have an IoU of larger than 0.67, 0.58, 0.53, 0.56, 0.63, and 0.68 respectively. To provide a general idea of how accurate the predicted bounding boxes represented by such values of IoU, we calculate the IoU for the three predicted bounding boxes in Figure 2. For the GW, Low frequency line and Blip, the IoUs are



(a)



(b)

Figure 8. The match between the predicted bounding boxes by `GSpyNetTreeS` and the ground truths for the testing samples, quantified by IoU. The upper panel shows the distribution of IoU for each of the classes of region of excess power. The lower panel shows the cumulative distributions. The red dashed horizontal line includes 90% of the testing samples.

0.73, 0.85 and 0.73 respectively. Since the ground truth bounding boxes in the training samples are manually drawn, inherent variations not due to the regions of excess power or the training process are expected and will be reflected in the results.

4. CONCLUSION

In this paper, as a proof of concept, we demonstrated `GSpyNetTreeS`, a novel machine learning algorithm based on YOLO, for the automatic identification, localization in the time and frequency domain, and classification of glitches that is robust to any GWs present. To train `GSpyNetTreeS`, we used glitches frequently observed by the LIGO detectors including Blip, Low frequency blip, Low frequency line, Scratchy, and Scattering, and simulated GWs added to real data collected during O3.

We demonstrated the performance of `GSpyNetTreeS` using metrics such as precision, recall and IoU. We showed that `GSpyNetTreeS` performs well for GW, Blip and Low frequency blip and reasonably well for Scratchy and Low frequency line. The poor performance for Scattering suggested insufficient training samples for this type of glitches. Using the metric IoU, we showed that for the regions of excess power that are correctly identified, `GSpyNetTreeS` can provide information such the frequency ranges and time windows that are consistent and representative of the ground truth regions of excess power. This suggests that `GSpyNetTreeS` can offer useful information for event validation and minimize human interference in the current event validation process employed by the LVK Collaboration, making results consistent and reproducible across different GW events. This approach is complementary to the more recently developed `GW-YOLO`, a segmentation algorithm that outputs pixel localization of GWs and glitches (Soni et al. 2025a). `GSpyNetTreeS`, which leverages the multi-label capabilities of `GSpyNetTree` to differentiate between GWs and multiple types of glitches in the same image, offers finer-grained classification of identified glitches. Additionally, `GSpyNetTreeS`'s time-frequency box output is designed to plug directly into existing glitch mitigation tools (e.g. `BayesWave`) which search an identified region for coherent signal power and incoherent glitch power.

In the future, increasing the number of samples in training for Scattering and Low frequency line beyond what is currently available in the `GSpyNetTree` data set is expected to improve the performance of `GSpyNetTreeS` for these two types of glitches. Additionally, glitches with different characteristic time scales may not be fully captured by a time-frequency representation of one-second in duration. For this reason, image-based glitch classification pioneered by `Gravity Spy` (Zevin et al. 2017), including `GSpyNetTree` (Alvarez-Lopez et al. 2024), uses multiple time-frequency representations of different time-scales to provide the algorithms with more comprehensive information of the data (Wu et al. 2025). One natural step to improve `GSpyNetTreeS` therefore is to include multiple time-frequency representation images with different time-scales. Including other glitches often observed by ground-based detectors such as Tomte and Koi fish will also make the algorithm more useful and applicable. We expect that `GSpyNetTreeS` will be a significant advance in automating LIGO-Virgo-

KAGRA's event validation process for the next observing run, allowing the collaboration to maintain its high standard of results vetting with an increased expected rate of candidate GW detections.

ACKNOWLEDGMENTS

We would like to express our gratitude to Beverly Berger for their insights that improved this work. We thank members of the ML-ESTEEM collaboration, in particular Daryl Haggard, Ashish Mahabal, Audrey Durand, Renée Hložek, Flavie Lavoie-Cardinal, Nayyer Raza, Alexandre Larouche, Hadi Moazen, and Frédéric Beaupré for helpful comments and discussions. J. M. acknowledges support from the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant program, the Canada Research Chairs (CRC) program, and IUSSTF (JC-001/2017). This material is based upon work supported by NSF's LIGO Laboratory, which is a major facility fully funded by the National Science Foundation.

REFERENCES

Aasi, J., et al. 2015, Class. Quant. Grav., 32, 074001, doi: 10.1088/0264-9381/32/7/074001

Abbott, B. P., Abbott, R., Abbott, Collaboration, T. L. S., & the Virgo Collaboration. 2020, Classical and Quantum Gravity, 37, 055002, doi: 10.1088/1361-6382/ab685e

Abbott, B. P., Abbott, R., Abbott, T. D., et al. 2016, Phys. Rev. Lett., 116, 061102, doi: 10.1103/PhysRevLett.116.061102

Abbott, B. P., et al. 2019, Phys. Rev. X, 9, 031040, doi: 10.1103/PhysRevX.9.031040

Abbott, R., et al. 2021a, Phys. Rev. X, 11, 021053, doi: 10.1103/PhysRevX.11.021053

—. 2021b, Phys. Rev. D, 104, 122004, doi: 10.1103/PhysRevD.104.122004

—. 2022a, Phys. Rev. D, 106, 102008, doi: 10.1103/PhysRevD.106.102008

—. 2023, Phys. Rev. X, 13, 041039, doi: 10.1103/PhysRevX.13.041039

Abbott, T. C., Buffaz, E., Vieira, N., et al. 2022b, The Astrophysical Journal, 927, 232, doi: 10.3847/1538-4357/ac5019

Acernese, F., et al. 2015, Class. Quant. Grav., 32, 024001, doi: 10.1088/0264-9381/32/2/024001

Alif, M. A. R., & Hussain, M. 2024, arXiv preprint arXiv:2406.10139

Alléné, C., Aubin, F., Bentara, I., et al. 2025, Classical and Quantum Gravity, 42, 105009, doi: 10.1088/1361-6382/add234

Alvarez-Lopez, S., Liyanage, A., Ding, J., Ng, R., & McIver, J. 2024, Class. Quant. Grav., 41, 085007, doi: 10.1088/1361-6382/ad2194

Areeda, J. S., Smith, J. R., Lundgren, A. P., et al. 2017, Astron. Comput., 18, 27, doi: 10.1016/j.ascom.2017.01.003

Aso, Y., Michimura, Y., Somiya, K., et al. 2013, Phys. Rev. D, 88, 043007, doi: 10.1103/PhysRevD.88.043007

Berger, B. K. 2018, Journal of Physics: Conference Series, 957, 012004, doi: 10.1088/1742-6596/957/1/012004

Biswas, R., Blackburn, L., Cao, J., et al. 2013, Phys. Rev. D, 88, 062003, doi: 10.1103/PhysRevD.88.062003

Cabero, M., Mahabal, A., & McIver, J. 2020, Astrophys. J. Lett., 904, L9, doi: 10.3847/2041-8213/abc5b5

Canton, T. D., Bhagwat, S., Dhurandhar, S. V., & Lundgren, A. 2014, Class. Quant. Grav., 31, 015016, doi: 10.1088/0264-9381/31/1/015016

Chan, M. L., Heng, I. S., & Messenger, C. 2020, Phys. Rev. D, 102, 043022, doi: 10.1103/PhysRevD.102.043022

Chan, M. L., McIver, J., Mahabal, A., et al. 2024, The Astrophysical Journal, 972, 50, doi: 10.3847/1538-4357/ad496a

Chatterjee, C., Kovalam, M., Wen, L., et al. 2023, Astrophys. J., 959, 42, doi: 10.3847/1538-4357/ad08b7

Chatterji, S., Blackburn, L., Martin, G., & Katsavounidis, E. 2004, Class. Quant. Grav., 21, S1809, doi: 10.1088/0264-9381/21/20/024

Cheng, T., Song, L., Ge, Y., et al. 2024, YOLO-World: Real-Time Open-Vocabulary Object Detection. https://arxiv.org/abs/2401.17270

Chu, Q., Kovalam, M., Wen, L., et al. 2022, Phys. Rev. D, 105, 024023, doi: 10.1103/PhysRevD.105.024023

Cornish, N. J., & Littenberg, T. B. 2015, Classical and Quantum Gravity, 32, 135012, doi: 10.1088/0264-9381/32/13/135012

Cuoco, E., et al. 2021, Mach. Learn. Sci. Tech., 2, 011002, doi: 10.1088/2632-2153/abb93a

Dal Canton, T., Nitz, A. H., Gadre, B., et al. 2021, The Astrophysical Journal, 923, 254, doi: 10.3847/1538-4357/ac2f9a

Davis, D., Littenberg, T. B., Romero-Shaw, I. M., et al. 2022, Class. Quant. Grav., 39, 245013, doi: 10.1088/1361-6382/aca238

Davis, D., Areeda, J. S., Berger, B. K., et al. 2021a, Classical and Quantum Gravity, 38, 135014, doi: 10.1088/1361-6382/abfd85

Davis, D., et al. 2021b, Class. Quant. Grav., 38, 135014, doi: 10.1088/1361-6382/abfd85

Essick, R., Blackburn, L., & Katsavounidis, E. 2013, Classical and Quantum Gravity, 30, 155010, doi: 10.1088/0264-9381/30/15/155010

Ewing, B., Huxford, R., Singh, D., et al. 2024, Phys. Rev. D, 109, 042008, doi: 10.1103/PhysRevD.109.042008

Gabbard, H., Messenger, C., Heng, I. S., Tonolini, F., & Murray-Smith, R. 2022, Nature Phys., 18, 112, doi: 10.1038/s41567-021-01425-7

Gabbard, H., Williams, M., Hayes, F., & Messenger, C. 2018, Phys. Rev. Lett., 120, 141103, doi: 10.1103/PhysRevLett.120.141103

Ghonge, S., Brandt, J., Sullivan, J. M., et al. 2024, Phys. Rev. D, 110, 122002, doi: 10.1103/PhysRevD.110.122002

Glanzer, J., Banagari, S., Coughlin, S., et al. 2021, Gravity Spy Machine Learning Classifications of LIGO Glitches from Observing Runs O1, O2, O3a, and O3b, v1.0.0, Zenodo, doi: 10.5281/zenodo.5649212

Green, S. R., Simpson, C., & Gair, J. 2020, Phys. Rev. D, 102, 104057, doi: 10.1103/PhysRevD.102.104057

Heaton, J. 2018, Genetic Programming and Evolvable Machines, 19, 305

Hourihane, S., Chatziioannou, K., Wijngaarden, M., et al. 2022, Phys. Rev. D, 106, 042006, doi: 10.1103/PhysRevD.106.042006

Husa, S., Khan, S., Hannam, M., et al. 2016, Phys. Rev. D, 93, 044006, doi: 10.1103/PhysRevD.93.044006

Jarov, S., Thiele, S., Soni, S., et al. 2023. https://arxiv.org/abs/2307.15867

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. 2022, Procedia Computer Science, 199, 1066, doi: https://doi.org/10.1016/j.procs.2022.01.135

Khan, S., Husa, S., Hannam, M., et al. 2016, Phys. Rev. D, 93, 044007, doi: 10.1103/PhysRevD.93.044007

Khanam, R., & Hussain, M. 2024, YOLOv11: An Overview of the Key Architectural Enhancements. https://arxiv.org/abs/2410.17725

Koyama, N., Sakai, Y., Sasaoka, S., et al. 2024, Machine Learning: Science and Technology, 5, 035028, doi: 10.1088/2632-2153/ad6391

Langendorff, J., Kolmus, A., Janquart, J., & Van Den Broeck, C. 2023, Phys. Rev. Lett., 130, 171402, doi: 10.1103/PhysRevLett.130.171402

LIGO Scientific Collaboration. 2018, LIGO Algorithm Library - LALSuite, free software (GPL), doi: 10.7935/GT1W-FZ16

Macas, R., Pooley, J., Nuttall, L. K., et al. 2022, Phys. Rev. D, 105, 103021, doi: 10.1103/PhysRevD.105.103021

Mishra, T., et al. 2022, Phys. Rev. D, 105, 083018, doi: 10.1103/PhysRevD.105.083018

Murali, C., & Lumley, D. 2023, Phys. Rev. D, 108, 043024, doi: 10.1103/PhysRevD.108.043024

Pankow, C., Chatziioannou, K., Chase, E. A., et al. 2018, Phys. Rev. D, 98, 084016, doi: 10.1103/PhysRevD.98.084016

Payne, E., Hourihane, S., Golomb, J., et al. 2022, Phys. Rev. D, 106, 104017, doi: 10.1103/PhysRevD.106.104017

Powell, J. 2018, Class. Quant. Grav., 35, 155017, doi: 10.1088/1361-6382/aacf18

Powell, J., Sun, L., Gereb, K., Lasky, P. D., & Dollmann, M. 2023, Class. Quant. Grav., 40, 035006, doi: 10.1088/1361-6382/acb038

Raza, N., Chan, M. L., Haggard, D., et al. 2024, Astrophys. J., 963, 98, doi: 10.3847/1538-4357/ad13ea

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. 2016, You Only Look Once: Unified, Real-Time Object Detection. https://arxiv.org/abs/1506.02640

Redmon, J., & Farhadi, A. 2018, YOLOv3: An Incremental Improvement. https://arxiv.org/abs/1804.02767

Rezatofighi, H., Tsoi, N., Gwak, J., et al. 2019, Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. https://arxiv.org/abs/1902.09630

Skliris, V., Norman, M. R. K., & Sutton, P. J. 2024, Phys. Rev. D, 110, 104034, doi: 10.1103/PhysRevD.110.104034

Soni, S., Mukund, N., & Katsavounidis, E. 2025a. https://arxiv.org/abs/2508.17399

Soni, S., Berger, B. K., Davis, D., et al. 2025b, Classical and Quantum Gravity, 42, 085016, doi: 10.1088/1361-6382/adc4b6

Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. 2023, Machine Learning and Knowledge Extraction, 5, 1680–1716, doi: 10.3390/make5040083

Tsukada, L., Joshi, P., Adhicary, S., et al. 2023, Phys. Rev. D, 108, 043004, doi: 10.1103/PhysRevD.108.043004

Udall, R., Hourihane, S., Miller, S., et al. 2025, Phys. Rev. D, 111, 024046, doi: 10.1103/PhysRevD.111.024046

Usman, S. A., Nitz, A. H., Harry, I. W., et al. 2016, Classical and Quantum Gravity, 33, 215004, doi: 10.1088/0264-9381/33/21/215004

Vazsonyi, L., & Davis, D. 2023, Class. Quant. Grav., 40, 035008, doi: 10.1088/1361-6382/acafd2

Wang, C.-Y., Liao, H.-Y. M., et al. 2024, APSIPA Transactions on Signal and Information Processing, 13

Wu, Y., Zevin, M., Berry, C. P. L., et al. 2025, Classical and Quantum Gravity, 42, 165015, doi: 10.1088/1361-6382/adf58b

Zevin, M., et al. 2017, Class. Quant. Grav., 34, 064003, doi: 10.1088/1361-6382/aa5cea

—. 2024, Eur. Phys. J. Plus, 139, 100, doi: 10.1140/epjp/s13360-023-04795-4