LARGE SCALE RADIO FREQUENCY WIDEBAND SIGNAL DETECTION & RECOGNITION

Luke Boegner[1], Garrett Vanhoy[1], Phillip Vallance[2], Manbir Gulati[3], Dresden Feitzinger[1], Bradley Comar[2], and Robert D. Miller[1]

[1]Peraton Labs
[2]Laboratory for Telecommunication Sciences
[3]Applied Insight

{*luke.boegner,gvanhoy,dresden.feitzinger,rmiller*}*@peratonlabs.com*
{*pvallance,bcomar*}*@ltsnet.net*

ABSTRACT

Applications of deep learning to the radio frequency (RF) domain have largely concentrated on the task of narrowband signal classification after the signals of interest have already been detected and extracted from a wideband capture. To encourage broader research with wideband operations, we introduce the Wideband-Sig53 (WBSig53) dataset which consists of 550 thousand synthetically-generated samples from 53 different signal classes containing approximately 2 million unique signals. We extend the TorchSig signal processing machine learning toolkit for open-source and customizable generation, augmentation, and processing of the WB-Sig53 dataset. We conduct experiments using state of the art (SoTA) convolutional neural networks and transformers with the WBSig53 dataset. We investigate the performance of signal detection tasks, i.e. detect the presence, time, and frequency of all signals present in the input data, as well as the performance of signal recognition tasks, where networks detect the presence, time, frequency, and modulation family of all signals present in the input data. Two main approaches to these tasks are evaluated with segmentation networks and object detection networks operating on complex input spectrograms. Finally, we conduct comparative analysis of the various approaches in terms of the networks' mean average precision, mean average recall, and the speed of inference[1].

1 INTRODUCTION

Radio frequency machine learning (RFML) has brought about innovative solutions to the problem of modulation classification (O'Shea & West, 2016), (O'Shea et al., 2018), (Boegner et al., 2022). These efforts provide a foundation for training ML models to classify signals in an environment where the center frequency and bandwidth of the signal is known *a-priori* but do not address the significantly more difficult context of a wideband environment. In a wideband environment, a signal must first be detected, localized in the time-frequency domain, and then finally classified. What makes this problem more difficult than the narrowband case is the presence of many other potentially interfering signals, receiver impairments such as dynamic range that are not as prominent in a narrowband environment, and the sheer amount of data to be processed. Attempting to detect and classify potentially thousands of signal bursts per second coming from numerous types of receivers in real-time remains an open engineering problem even when considering classical detection methods. A key limiter of more research in this area is the lack of openly available datasets by which researchers can more objectively compare their results. In this work, we introduce WBSig53, a benchmark dataset that includes many peculiarities of the wideband signal detection problem and initial results using state of the art (SoTA) deep learning models adapted to the RF domain.

---

[1]The TorchSig toolkit is available at https://github.com/torchdsp/torchsig with additional documentation available at https://torchsig.com. The WBSig53 dataset can be generated using the TorchSig toolkit.

Section 2 of this paper discusses prior RFML work and datasets for signal detection and recognition. Section 3 introduces the WBSig53 dataset and the TorchSig RFML software toolkit. Section 4 describes the approaches and neural networks used in the experiments conducted below. Section 5 shares initial experimentation and performance of various neural networks on the WBSig53 dataset for the task of signal detection, while Section 6 addresses the task of signal recognition. The paper then concludes with Section 7.

2 PRIOR WORK

In Wong et al. (2021), researchers provide an excellent overview of the complexities of the RFML ecosystem. In particular, they describe the major areas of overlap and voids in the broader RFML research space. Notably, they highlight the importance of *data* towards learned behavior and describe the intricacies of open and custom RFML datasets.

In Vagollari et al. (2021), wideband spectrograms are used to perform detection, localization, and classification. Object detection is performed via YOLO adaptations using simulated datasets that cover analog and digital modulations. The classes are somewhat limited, and the authors group PSK and QAM into the PAM class due to their (1) reliance on spectrograms as the network input and (2) spectrogram similarity. The researchers do include some channel variations, albeit in a limited fashion.

As part of the IEEE SPAWC2021 Challenge, researchers developed and released a wideband dataset (West et al., 2021a) to spur competition and promote research in this area. The dataset was synthetically generated, covering 130 unique band layouts and 14 modulations with some emulated real-world channel effects. West et al. (2021b) describes the dataset and also presents and analyzes a U-Net approach to the problem (Ronneberger et al., 2015).

Another wideband dataset was discussed in Nguyen et al. (2021) covering emissions from real devices using five signal protocols (Bluetooth, Lightbridge, Wi-Fi, XPD, and Zigbee). The data was collected at 100 MHz and expertly labeled, yielding 1.4 TBytes of RFML data for the community. A generalized YOLO approach was also presented.

3 WIDEBANDSIG53 DATASET

The WidebandSig53 (WBSig53) dataset is envisioned to facilitate the advancement of SoTA signal detection and recognition techniques in the RFML domain. Given the vastly diverse nature of signal types, wireless environments, and transmitter/receiver impairments in the real world, it is impossible to fully capture all possible distributions within a tractable dataset. However, the WBSig53 dataset incorporates many of the significant, representative challenges observed during the practical application of signal detection and recognition. When a model performs well on the WBSig53 dataset, ideally, it should also be able to successfully transfer-learn to a real world solution. This concept mirrors how computer vision object detection and segmentation models can be pre-trained on the COCO dataset (Lin et al., 2014) and serve as a solid foundation for future, specific applications. WBSig53 fills a gap in this research area by providing the following:

Open-Source Dataset Generation: While several efforts highlighted in Section 2 introduced wideband signals datasets, they focus on specific applications and provide limited scale, flexibility, and/or reproducibility. By introducing WBSig53 as a synthetically generated dataset, we provide the community with a common benchmark for collaborative exploration in a large scale manner that is easily *extensible*, *reviewable*, and *reproducible*.

Large Signal Diversity: Many works in the wideband signal detection and recognition space have a limited number of signal classes. While this is likely intentional due to the complexity of signal recognition tasks, we provide 53 unique signal classes in the WBSig53 dataset to maximize future research potential. We also provide target transforms within the TorchSig toolkit such that the difficult task of signal recognition over 53 classes can be easily mapped down to fewer classes by means of modulation family groupings or all the way down to a signal detection task with a single "signal" class.

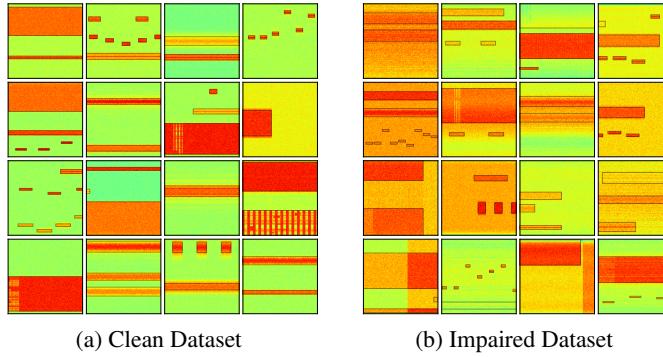(a) Clean Dataset   (b) Impaired Dataset

Figure 1: Spectrograms of examples from the clean and impaired WBSig53 datasets.

Impairment Diversity: Past work in Miller et al. (2019), Scholl (2022), and Boegner et al. (2022) have all shown the benefits of synthetic impairments to RFML training data. During the WBSig53 dataset generation, the complex-valued samples are impaired by applying a diversified subset of 11 emulated real world RF impairments.

The WBSig53 dataset consists of 550 thousand examples containing $\sim$2 million signals split into 4 distinct sub-datasets: *Clean Training* (250k examples), *Clean Validation* (25k examples), *Impaired Training* (250k examples), and *Impaired Validation* (25k examples).

The WBSig53 sub-datasets split training and validation examples, such that the validation sets can be used collaboratively and competitively across the RFML community. This is similar to the computer vision community's use of the COCO dataset (and corresponding challenge) as a benchmark. The WBSig53 dataset also provides training and validation datasets in both clean and impaired formats. Impaired examples are passed through realistic RF impairments as defined below, while the clean examples enable researchers to apply customized impairments tailored to their use case.

The default example size across all sub-datasets is $262,144$ complex-valued samples, representing I (in-phase or real) and Q (quadrature or imaginary) samples. This size enables the use of a 512-point FFT operation (with similar step size for no overlap) in the creation of square 512×512 complex-valued spectrograms. This size spectrogram can be used with many modern ML techniques used in the vision domain while still encompassing a large amount of relevant signal data.

3.1 CLEAN WBSIG53 DATASET

The clean datasets randomly select 1 to 6 signal sources. Each signal source is randomly given a signal-to-noise ratio (SNR) in the range of 20 to 40dB E_s/N_0 (energy per symbol to noise power spectral density). The type of signal for each source is randomly selected from the full list of signal classes in Table 3. The modulation families within the list include: amplitude-shift keying (ASK), pulse-amplitude modulation (PAM), phase-shift keying (PSK), quadrature amplitude modulation (QAM), frequency-shift keying (FSK), and orthogonal frequency-division multiplexing (OFDM).

3.2 IMPAIRED WBSIG53 DATASET

The impaired datasets impose synthetic impairments emulating environmental effects and real-world system impairments. Prior to any impairments, signal sources' SNRs change to be uniformly distributed between 0 and 30db E_s/N_0.

The following 5 impairments are applied to full data examples with parenthetically specified likelihoods[2]: *Time Shift* (25%), *Frequency Shift* (25%), *Random Resampling* (25%), *Spectral Inversion* (50%), and *Additive White Gaussian Noise* (100%).

---

[2]In wideband multi-signal scenarios, there are typically different transmitter and wireless channel impairments for different signal sources. We currently limit the impairments to occur on the full data example, affecting all signal sources equally. We suspect this still encompasses a sufficiently challenging task and models trained using this subset of realism should still provide an equally valuable baseline for transfer learning.

To introduce additional impairment diversity, we also use a modified RandAugment transform (Cubuk et al., 2020). Our RandAugment transform inputs 7 additional impairments and randomly selects 2 of them to apply for each data example. These 7 additional impairments are: *Magnitude Rescaling*, *RF Roll-Off*, *Random Convolve*, *Rayleigh Fading Channel*, *Drop Samples*, *Phase Shift*, and *IQ Imbalance*.

Spectrograms of a subset of WBSig53 clean and impaired data can be seen in Figure 1. More details and visualizations of these impairments and additional augmentations are shown in Appendix A.1 and Appendix A.2, respectively.

4 Neural Networks

Signal presence detection and localization (i.e. time and/or frequency estimation) have been explored using signal processing-based techniques, such as in Spooner (2007), Olivieri et al. (2005), and Vartiainen et al. (2010). A more comprehensive survey is discussed in Yucek & Arslan (2009). While there exists opportunities to explore signal processing-based techniques using the WBSig53 dataset, our work is scoped to the exploration of neural networks' signal detection and recognition abilities.

We construct the tasks of signal detection and recognition with an object detection and a segmentation approach. Within each approach, we experiment with one neural architecture that is convolutional-based and one that is transformer-based (Vaswani et al., 2017).

4.1 Object Detection

Object detection algorithms are a class of ML techniques that directly infer bounding boxes and classes given an input image. These algorithms lend themselves well to signal detection and recognition tasks because most signals (and all signals currently within the WBSig53 dataset) are rectangular in time and frequency. Within this class of algorithms, we choose to explore YOLOv5 (Jocher et al., 2022) as a fast, convolutional-based approach and DETR (Carion et al., 2020) as a more powerful, transformer-based approach.

With YOLOv5, we experiment with the YOLOv5-nano and YOLOv5-small scales. We also introduce a further scaled down version, which we term YOLOv5-pico. With DETR, we modify the architecture slightly by using EfficientNet (Tan & Le, 2019) backbones and an XCiT transformer (El-Nouby et al., 2021). We make these changes for data efficiency and increased model inference speed. For all DETR experiments, we use the smallest scale XCiT-nano. For the backbones, we vary scale by using EfficientNet-B0, B2, and B4. We term these models DETR-B0-Nano, DETR-B2-Nano, and DETR-B4-Nano, respectively.

4.2 Segmentation

Segmentation algorithms are a class of ML techniques that infer object instance and/or class associations pixel-by-pixel in an input image. Within segmentation algorithms, there are three common subcategories: semantic, instance, and panoptic segmentation. Semantic segmentation infers pixel-by-pixel class labels, where overlapping classes are left ambiguous. Instance segmentation infers pixel-by-pixel object instance labels, where overlapping objects of the same class are disambiguated; however, there is no class information associated with each instance. Panoptic segmentation combines semantic and instance segmentation, where all instances and class information for each instance are inferred. In the WBSig53 dataset, there are no overlapping signals in both time and frequency, so we focus on the task of semantic segmentation for both signal detection and recognition. We impose a naive post-processing step that converts the masks to bounding boxes (Figure 2). This allows us to report signals in familiar time-frequency values and easily compare segmentation models to object detection models. Within this class of algorithms, we explore PSPNet (Zhao et al., 2016) as a convolutional-based approach, and Mask2Former (Cheng et al., 2021a) as a more powerful, transformer-based approach.

With PSPNet, we modify the original architecture slightly by using EfficientNet as our encoders at three distinct scales: EfficientNet-B0, B2, and B4. We term these models PSPNet-B0, PSPNet-B2, and PSPNet-B4, respectively. Similarly with Mask2Former, we use EfficientNet backbones at the same scales. We term these models Mask2Former-B0, Mask2Former-B2, and Mask2Former-B4, respectively.
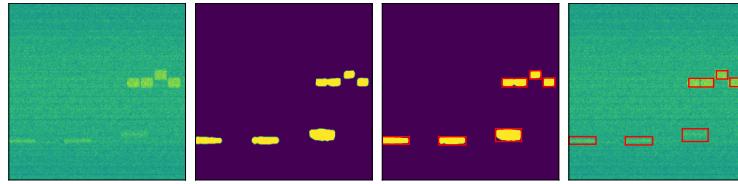
Figure 2: Segmentation masks are converted to boxes for signal predictions.

5   SIGNAL DETECTION EXPERIMENTS

In this section, we present results using the previously introduced neural networks to perform signal detection on the WBSig53 impaired dataset. We define signal detection as the ability of an algorithm to detect the presence of any and all signals within an input sample, as well as discern the location in time and frequency of each signal. This is analogous to object detection in the vision domain, where instead of inferring (x, y)-coordinates of objects within an image, we infer (t, f)-coordinates of signals within a data capture.

To better leverage research from the vision domain, we frame our tasks as closely to object detection as possible by applying a complex-valued spectrogram transform to WBSig53's complex-valued IQ samples. This transform is applied with an FFT size of 512, a segment length of 512, no overlap, and a Blackman-Harris window, resulting in a critically sampled complex-valued spectrogram of size $(512, 512)$. As a final step, the complex values are split into real and imaginary components and concatenated in the channel dimension, similar to how RGB pixels are handled in the vision domain. The resulting shape of our input samples is then $(2, 512, 512)$. With this shape, many vision-based networks can be leveraged with a simple modification to the channel dimension from 3 (RGB) down to 2 (real/imaginary).

In addition to the spectrogram data transform, we also apply a target transform that maps the WBSig53 signal labels to each neural networks' expected target format. In our signal detection experiments, these target transforms map all 53 signal classes contained within WBSig53 to a single "signal" class. In the object detector approaches (DETR and YOLOv5), the transformed targets are bounding box labels. In the vision domain, these bounding box labels are often represented as (x_c, y_c, w, h), where x_c represents the center x position, y_c represents the center y position, w represents the width of the box, and h represents the height of the box. We follow a similar convention; however, our bounding box labels are represented as (t_c, f_c, d, B), where t_c represents the center time, f_c represents the center frequency, d represents the signal duration, and B represents the signals' bandwidth. In the case of the segmentation approaches (PSPNet and Mask2Former), the transformed targets are semantic masks where all areas containing signals in the input sample are labeled with 1 and all areas outside these signals are labeled with 0, forming a pixel-by-pixel label of all signal instances as masks within a $(512, 512)$-sized matrix.

For evaluation, we adopt the Mean Average Precision (mAP) metric. mAP is a widely-used metric within object detection tasks in the vision domain due to its ability to quantify performance across classification and localization, while jointly measuring precision and recall across varying detection criteria, namely a discrete set of intersection over union (IoU) thresholds. In addition to the mAP score, we also report AP_{50}, AP_{75}, AP_S, AP_M, and AP_L, as defined in the TorchMetrics toolkit (Skafte et al., 2020). We also inspect the Mean Average Recall (mAR) score. mAR measures an algorithm's ability to detect all instances within an input sample, where a target object is considered detected if a predicted box's IoU score meets a given threshold. While the mAR score is often omitted in vision domain evaluations, we include it because of its similarities with a signal detection rate that is more commonly used in the signals domain.

All networks are pulled from their original authors' PyTorch implementations and then modified as discussed previously (Paszke et al., 2017). Despite the benefits of data augmentations, we limit our experiments to the static data samples found directly in the WBSig53 impaired dataset. Within each network architecture, we train the various scales with identical training parameters; however, we vary parameters across architectures. All networks are trained on a single Nvidia V100 GPU, lasting from days to weeks depending on network speeds.

Table 1: Signal detection results for various networks using the WBSig53 impaired dataset. YOLOv5-pico is the fastest, while DETR-B4-Nano is the best performer. Throughput is measured in samples per second (SPS) on a single Nvidia V100 GPU.

Model	Params	SPS	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	mAR
YOLOv5-pico	0.32 M	1335.21 M	73.03	87.00	78.81	67.28	77.56	68.64	75.17
YOLOv5-nano	1.8 M	796.35 M	73.82	86.99	79.97	68.82	78.29	69.29	75.92
YOLOv5-small	7.0 M	432.61 M	73.64	86.98	79.94	68.86	78.09	68.92	75.78
DETR-B0-Nano	8.2 M	161.21 M	85.49	97.93	92.24	73.67	84.62	90.92	89.62
DETR-B2-Nano	11.9 M	119.17 M	85.83	97.93	92.22	74.47	85.12	90.98	90.06
DETR-B4-Nano	21.9 M	74.70 M	86.98	98.92	93.35	75.99	86.01	91.54	90.90
PSPNet-B0	4.1 M	215.54 M	68.52	92.65	73.54	39.32	62.66	86.34	70.89
PSPNet-B2	7.8 M	155.07 M	72.30	93.76	77.37	44.42	68.01	88.18	74.44
PSPNet-B4	17.6 M	97.03 M	73.59	93.83	79.42	46.79	69.68	88.81	75.79
Mask2Former-B0	22.2 M	19.28 M	71.97	84.48	78.89	62.12	65.75	84.13	78.86
Mask2Former-B2	26.0 M	18.47 M	77.29	90.77	84.98	61.76	73.30	92.40	82.45
Mask2Former-B4	36.3 M	16.87 M	81.05	94.31	89.25	64.16	78.64	91.15	85.35

YOLOv5 models are trained for 128 epochs with a batch size of 32. We use the Adam optimizer (Kingma & Ba, 2014) with a weight decay of $1e^{-4}$ and initialize the learning rate to $2e^{-3}$ while using a cosine annealing scheduler (Loshchilov & Hutter, 2016). DETR models are trained for 1M steps with a batch size of 32. We use the AdamW optimizer (Loshchilov & Hutter, 2017) with a weight decay of 0.04 and initialize the learning rate to $5e^{-4}$ while using a cosine annealing scheduler with 20k warm-up steps. PSPNet models are trained for 128 epochs with a batch size of 32. We use the Adam optimizer with a weight decay of $1e^{-4}$ and initialize the learning rate to $2e^{-3}$ while using a cosine annealing scheduler. Mask2Former models are trained for 1M steps with a batch size of 16. We use the AdamW optimzer with a weight decay of 0.04 and initialize the learning rate to $5e^{-4}$ while using a cosine annealing scheduler with 20k warm-up steps.

During training, we monitor the validation loss and save the lowest score for evaluation. For metric calculations, we use TorchMetrics for all models' inferred results on the WBSig53 impaired validation set. YOLOv5 and DETR models output bounding boxes with probabilities directly, and as such, their outputs undergo very minor processing prior to the metric computation. Mask2Former outputs masks with probabilities, so simply converting the masks to boxes with their associated probabilities is sufficient processing prior to computing the metrics. PSPNet only outputs masks, so in addition to the mask to box conversion, a probability is assigned as the average of every pixel-level probability within the box region.

Under our training settings, DETR reports the best performance in terms of both mAP and mAR. YOLOv5-pico achieves modest performance at significant speeds relative the other networks. Interestly, PSPNet reports the lowest mAP score; however, its AP_{50} score is greater than YOLOv5. This may indicate PSPNet is better at coarse detection than YOLOv5, but its localization suffers; this could
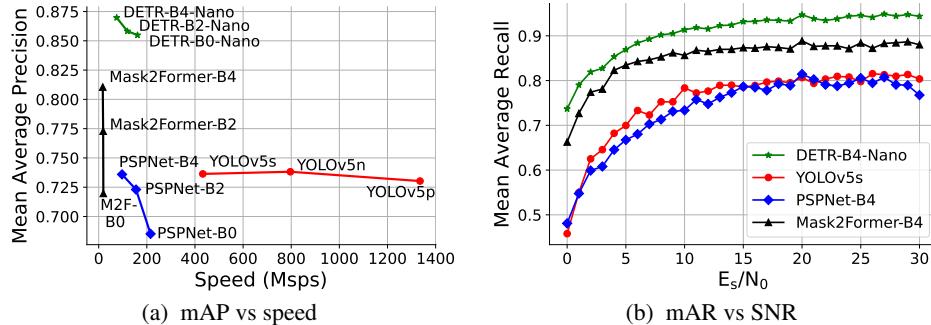

(a) mAP vs speed

(b) mAR vs SNR

Figure 3: Signal detection results. (a) mean average precision vs speed shows that DETR is the best performer while YOLOv5 is the fastest network. (b) mean average recall versus SNR shows that DETR performs best across all SNRs.
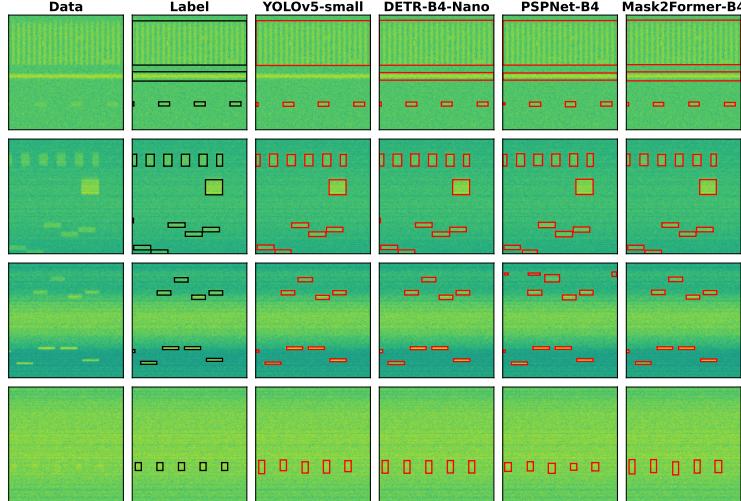
Figure 4: Signal detection model comparisons. DETR-B4-Nano is the only network to detect all signals due to a very small signal at the start of the second data sample. YOLOv5-small misses an addition signal in the first data example, despite the signal having fairly high SNR. PSPNet-B4 contains a few false positives in the third data example. Mask2Former-B4 performs well, only missing the small signal at the start of the second data example.

be due to the naive mask to box conversion process, where errors at the pixel-level bias PSPNet's output detections to be larger, thus hurting its localization. It is also interesting that Mask2Former fails to achieve the best performance given its large parameter advantage over the other networks. But, the increased parameter complexity comes with additional hyperparameters, likely indicating that its performance has significant room for improvement via hyperarameter tuning.

We next analyze each networks' largest scale's performance across SNRs in Figure 3b. With this metric, we analyze the networks' abilities to detect signals across various SNRs. Interestingly, this metric displays mAR differences across all SNR values across all networks. In a traditional signal detection algorithm, there is typically high performance at high SNR with a drop at lower SNRs, where the location of the drop is the main shift in performance, i.e. a better detector would see its drop in performance occur at lower SNRs than a lower quality detector's drop at higher SNRs. These findings suggest that even at high SNRs, our lower quality detectors still only detect at a rate of roughly 80%. This difference could be due to our mAR metric's joint detection and localization measure, whereas the traditional detectors prioritize presence detection without locality.

In a final analysis of the signal detectors' performances, we retrieve four random data examples from the WBSig53 dataset and plot their spectrograms, labels, and inferred results across all architectures' largest scales (Figure 4). Here, we see DETR-B4-Nano detecting all target signals with only minor bandwidth errors. YOLOv5-small has a missed detection in the top data example, despite the signal having fairly high SNR, and it misses a very small signal at the beginning of the second example. PSPNet-B4 has multiple false positives in the third data example, and it also misses the very small signal at the start of the third example. Mask2Former performs well; however, it also misses the very small signal at the start of the third example.

6 SIGNAL RECOGNITION EXPERIMENTS

We now present signal recognition results on the WBSig53 impaired dataset. We define signal recognition as an algorithm's ability to discern the presence, location (in time and frequency), and class of any and all signals within an input example. Here, we simplify the task to perform recognition over modulation family classes, rather than the granular signal class contained within the WBSig53 dataset. With recognition over modulation families, we reduce the valid classes to the 6 modulation families: ASK, FSK, OFDM, PAM, PSK, and QAM. Signal recognition is analogous to the vision domain's object recognition task, where instead of inferring (x, y)-coordinates and the class of objects within an image, our algorithms infer (t, f)-coordinates and the class of signals within a data capture.

Table 2: Signal recognition results on modulation family classes using the WBSig53 impaired dataset. YOLOv5-pico is the fastest network while DETR-B4-Nano is the best performer.

Model	Params	SPS	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	mAR
YOLOv5-pico	0.32 M	1335.21 M	39.46	50.08	42.70	22.28	49.18	44.45	48.74
YOLOv5-nano	1.8 M	796.35 M	57.02	72.51	63.24	37.61	55.13	52.39	62.51
YOLOv5-small	7.0 M	432.61 M	58.50	72.37	64.29	38.35	56.68	54.27	64.35
DETR-B0-Nano	8.2 M	161.21 M	76.52	86.15	82.45	54.30	74.15	84.39	83.62
DETR-B2-Nano	11.9 M	119.17 M	78.95	88.92	85.65	57.50	76.77	86.89	84.74
DETR-B4-Nano	21.9 M	74.70 M	80.65	88.64	85.41	59.24	78.48	88.48	86.03
PSPNet-B0	4.1 M	215.54 M	27.30	39.07	28.29	07.07	21.47	41.85	44.50
PSPNet-B2	7.8 M	155.07 M	36.67	49.40	38.26	11.75	29.26	44.65	47.89
PSPNet-B4	17.6 M	97.03 M	51.84	67.23	56.54	16.85	47.59	68.53	62.61
Mask2Former-B0	22.2 M	19.28 M	14.78	17.68	15.23	04.20	11.35	15.85	38.19
Mask2Former-B2	26.0 M	18.47 M	22.01	25.80	23.29	09.20	16.96	25.27	45.18
Mask2Former-B4	36.3 M	16.87 M	27.03	32.13	29.22	08.23	22.85	33.38	52.81

We follow the same data processing pipeline as the detection task, and we update our models to output 6 modulation family classes, rather than the single "signal" class for detection. Here, AP and AR metrics now average across all classes in addition to all IoU thresholds. Once again, we train all networks using a single V100 GPU over the span of days to weeks, depending on model speed. We train the models with the static WBSig53 impaired training set without the use of data augmentations. We use the same training parameters and schedules as the detection experiments, except for the PSPNet models. The PSPNet models are trained for $1M$ steps with a batch size of 32. We use the AdamW optimizer with a weight decay of 0.01 and initialize the learning rate to $1e^{-3}$ while using a cosine annealing scheduler with $4k$ warm-up steps.

During training, we monitor the validation loss and save the best value for evaluation. We evaluate the best model using TorchMetrics and record the results in Table 2. Additionally, we display the mAP versus speed for each network in Figure 5a.

DETR again reports the highest performance in terms of both mAP and mAR. Interestingly, we see our YOLOv5-pico model takes a larger hit in performance compared to its larger scale, YOLOv5-nano. We also see a sizeable decrease in performance with the Mask2Former experiments. Once again, this is likely due to Mask2Former's large size and complexity, and we suspect performance will improve with additional training and hyperparameter tuning.

We next analyze each networks' performance across SNRs in Figure 5b. Once again, we see DETR-B4-Nano perform the best across all SNRs. YOLOv5-small slightly outperforms PSPNet-B4. Mask2Former-B4 shows the weakest performance across all SNRs.

We retrieve four random data examples from the WBSig53 dataset and plot their spectrograms, labels, and inferred results across all architectures' largest scales (Figure 6). We color the bounding boxes to
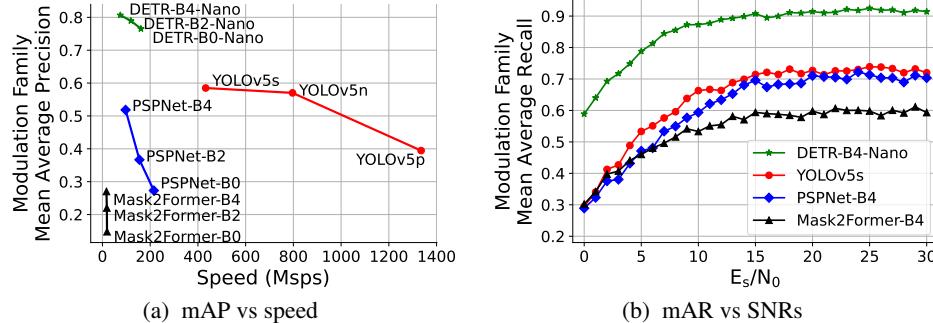

(a) mAP vs speed

(b) mAR vs SNRs

Figure 5: Signal recognition results across modulation families. (a) Mean average precision vs speed shows that DETR is the best performer while YOLOv5 is the fastest network. (b) Mean average recall versus SNR shows that DETR performs best across all SNRs.
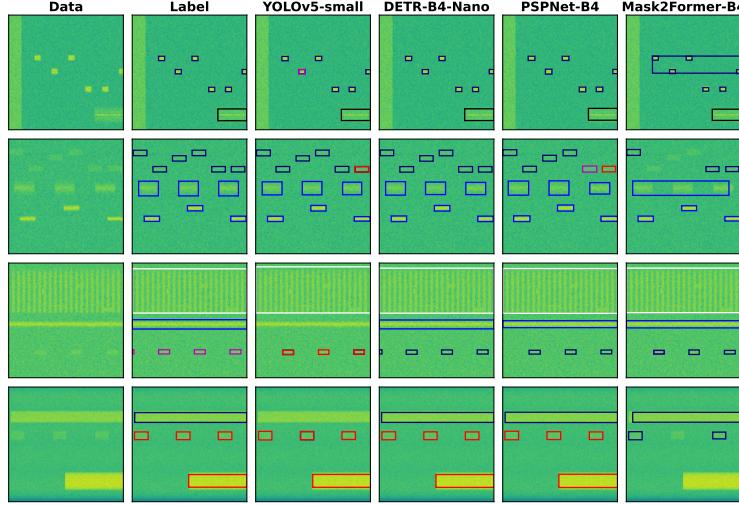
Figure 6: Signal recognition for modulation family classes model comparisons. We plot the modulation family label and prediction displayed as the color of the bounding boxes. DETR is the only network to detect all signals, but it misclassifies PSK signals as QAM in the third example. YOLOv5 suffers from missed detections of high SNR signals, and makes several misclassifications throughout. PSPNet detects all but one small signal at the start of the third example; however, it also suffers from misclassifications of several signals. Mask2Former struggles to make consistent, clean detections and classifications, likely due to non-optimal training parameters.

represent the modulation families, where ASK is red, FSK is blue, OFDM is white, PAM is black, PSK is magenta, and QAM is navy. DETR-B4-Nano is the only network to detect every signal, but it misclassifies a sequence of bursty PSK signals as QAM bursts in the third example. YOLOv5-small suffers from more misclassifications, and it fails to detect several signals despite high SNR. PSPNet-B4 detects all but a small signal at the start of the third data example; however, it suffers from misclassifying multiple signals throughout. Mask2Former-B4 shows relatively poor performance with missed detections, poor localizations, and misclassifications. We suspect Mask2Former's large complexity requires additional training optimizations in order to regain its performance advantage.

7 CONCLUSION

In this work, we highlight a gap in RFML research in signal detection and signal recognition algorithms due to a lack of datasets and tools to conduct this work. We introduce the WidebandSig53 dataset, which aims to fill this gap by providing an open-source benchmark enabling future RFML research. We share reproducible experimental results modifying SoTA scalable vision domain networks: YOLOv5, DETR, PSPNet, and Mask2Former. The application and thorough analysis of these networks marks one of the earliest applications of modern convolutional neural networks and transformers to the problems of large scale signal detection and signal recognition. We also extend the open-source toolkit, TorchSig, to be an easily usable tool for the generation, augmentation, and utilization of wideband signals datasets and models. While our work reports performances of multiple networks, we expect that these performances can improve with additional research, and we expect that other network architectures may outperform the limited subset explored in our work. We invite others to build upon our research by leveraging our work and open-source tools to demonstrate better performance than what we report here. We also invite others to investigate models capable of performing signal recognition across the full suite of fine-grain classes within the WBSig53 dataset. We hope the accessibility of this dataset and toolkit allows for broad collaborative advances in the field of wideband RFML research.

REFERENCES

Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. URL `https://arxiv.org/abs/2004.10934`.

Luke Boegner, Manbir Gulati, Garrett Vanhoy, Phillip Vallance, Bradley Comar, Silvija Kokalj-Filipovic, Craig Lennon, and Robert D. Miller. Large scale radio frequency signal classification, 2022. URL `https://arxiv.org/abs/2207.09918`. arXiv:2207.09918.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020. URL `https://arxiv.org/abs/2005.12872`.

Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation, 2021a. URL `https://arxiv.org/abs/2112.01527`.

Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021b.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Alaaeldin El-Nouby, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *arXiv preprint arXiv:2106.09681*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, February 2022. URL `https://doi.org/10.5281/zenodo.6222936`.

Guoliang Kang, Xuanyi Dong, Liang Zheng, and Yi Yang. Patchshuffle regularization, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pp. 83–97, 1955.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. URL `https://arxiv.org/abs/1405.0312`.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2016. URL `https://arxiv.org/abs/1608.03983`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Robert D. Miller, Silvija Kokalj-Filipovic, Garrett Vanhoy, and Joshua Morman. Policy based synthesis: Data generation and augmentation methods for rf machine learning. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1–5, 2019. doi: 10.1109/GlobalSIP45357.2019.8969160.

Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pp. 565–571. IEEE, 2016.

Hai N. Nguyen, Marinos Vomvas, Triet Vo-Huu, and Guevara Noubir. Wideband, real-time spectro-temporal rf identification. In *Proceedings of the 19th ACM International Symposium on Mobility Management and Wireless Access*, MobiWac '21, pp. 77–86, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450390798. doi: 10.1145/3479241.3486688. URL https://doi.org/10.1145/3479241.3486688.

Marc P Olivieri, Greg Barnett, Alex Lackpour, Albert Davis, and Phuong Ngo. A scalable dynamic spectrum allocation system with interference mitigation for teams of spectrally agile software defined radios. In *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, pp. 170–179. IEEE, 2005.

Timothy J O'Shea and Nathan West. Radio Machine Learning Dataset Generation with GNU Radio. In *Proceedings of the GNU Radio Conference*, volume 1, 2016.

Timothy James O'Shea, Tamoghna Roy, and T. Charles Clancy. Over-the-Air Deep Learning Based Radio Signal Classification. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):168–179, 2018. doi: 10.1109/JSTSP.2018.2797022.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016. URL https://arxiv.org/abs/1612.08242.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. URL https://arxiv.org/abs/1804.02767.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. URL https://arxiv.org/abs/1506.02640.

Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 658–666, 2019.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Stefan Scholl. Rf signal classification with synthetic training data and its real-world performance. *arXiv preprint arXiv:2206.12967*, 2022.

Nick Skafte, Jirka Borovec, and Justus Schock. Torchmetrics. https://github.com/Lightning-AI/metrics/blob/master/docs/source/index.rst, 2020.

Chad M Spooner. Multi-resolution white-space detection for cognitive radio. In *MILCOM 2007-IEEE Military Communications Conference*, pp. 1–9. IEEE, 2007.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Adela Vagollari, Viktoria Schram, Wayan Wicke, Martin Hirschbeck, and Wolfgang Gerstacker. Joint detection and classification of rf signals using deep learning. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pp. 1–7, 2021. doi: 10.1109/VTC2021-Spring51267. 2021.9449073.

Johanna Vartiainen, Janne Lehtomaki, Harri Saarnisaari, Markku Juntti, and Kenta Umebayashi. Two-dimensional signal localization algorithm for spectrum sensing. *IEICE transactions on communications*, 93(11):3129–3136, 2010.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Tailai Wen. tsaug. `https://github.com/arundo/tsaug`, 2019.

Nathan West, Timothy O'Shea, and Tamoghna Roy. A wideband signal recognition dataset. 2021a. doi: 10.48550/ARXIV.2110.00518. URL `https://arxiv.org/abs/2110.00518`.

Nathan West, Tamoghna Roy, and Timothy O'Shea. Wideband signal localization with spectral segmentation. 2021b. doi: 10.48550/ARXIV.2110.00583. URL `https://arxiv.org/abs/2110.00583`.

Lauren J. Wong, William H. Clark, Bryse Flowers, R. Michael Buehrer, William C. Headley, and Alan J. Michaels. An rfml ecosystem: Considerations for the application of deep learning to spectrum situational awareness. *IEEE Open Journal of the Communications Society*, 2:2243–2264, 2021. doi: 10.1109/OJCOMS.2021.3112939.

Tevfik Yucek and Huseyin Arslan. A survey of spectrum sensing algorithms for cognitive radio applications. *IEEE communications surveys & tutorials*, 11(1):116–130, 2009.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network, 2016. URL `https://arxiv.org/abs/1612.01105`.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

A APPENDIX

A.1 DATASET APPENDIX

Table 3: Full WBSig53 Class List

CLASS NAME	MODULATION FAMILY	CLASS INDEX
4ASK	ASK FAMILY	3
8ASK	ASK FAMILY	6
16ASK	ASK FAMILY	10
32ASK	ASK FAMILY	15
64ASK	ASK FAMILY	19
OOK (ON-OFF KEYING)	PAM FAMILY	0
4PAM	PAM FAMILY	2
8PAM	PAM FAMILY	5
16PAM	PAM FAMILY	9
32PAM	PAM FAMILY	14
64PAM	PAM FAMILY	18
BPSK (BINARY PHASE-SHIFT KEYING)	PSK FAMILY	1
QPSK (QUADRATURE PHASE-SHIFT KEYING)	PSK FAMILY	4
8PSK	PSK FAMILY	7
16PSK	PSK FAMILY	11
32PSK	PSK FAMILY	16
64PSK	PSK FAMILY	20
16QAM	QAM FAMILY	8
32QAM	QAM FAMILY	12
32QAM_Cross	QAM FAMILY	13
64QAM	QAM FAMILY	17
128QAM_Cross	QAM FAMILY	21
256AM	QAM FAMILY	22
512AM_Cross	QAM FAMILY	23
1024AM	QAM FAMILY	24
2FSK	FSK FAMILY	25
2GFSK	FSK FAMILY	26
2MSK	FSK FAMILY	27
2GMSK	FSK FAMILY	28
4FSK	FSK FAMILY	29
4GFSK	FSK FAMILY	30
4MSK	FSK FAMILY	31
4GMSK	FSK FAMILY	32
8FSK	FSK FAMILY	33
8GFSK	FSK FAMILY	34
8MSK	FSK FAMILY	35
8GMSK	FSK FAMILY	36
16FSK	FSK FAMILY	37
16GFSK	FSK FAMILY	38
16MSK	FSK FAMILY	39
16GMSK	FSK FAMILY	40
OFDM-64	OFDM FAMILY	41
OFDM-72	OFDM FAMILY	42
OFDM-128	OFDM FAMILY	43
OFDM-180	OFDM FAMILY	44
OFDM-256	OFDM FAMILY	45
OFDM-300	OFDM FAMILY	46
OFDM-512	OFDM FAMILY	47
OFDM-600	OFDM FAMILY	48
OFDM-900	OFDM FAMILY	49
OFDM-1024	OFDM FAMILY	50
OFDM-1200	OFDM FAMILY	51
OFDM-2048	OFDM FAMILY	52

Dataset Parameterization. The clean datasets randomly select a number of signal sources in the range of 1 to 6. Each signal source is randomly given an SNR in the range of 20dB to 40dB E_s/N_0 (energy per symbol to noise power spectral density). The signal source is randomly selected from the full list of signal classes seen in Table 3. The modulation families within the full class list include: amplitude-shift keying (ASK), pulse-amplitude modulation (PAM), phase-shift keying (PSK), quadrature amplitude modulation (QAM), frequency-shift keying (FSK), and orthogonal frequency-division multiplexing (OFDM).

OFDM signals are the only multi-carrier modulation family, and as such, we bias OFDM bandwidths to be larger than the single-carrier waveforms in examples that contain multiple signals of differing modulation families. While an OFDM signal may have a bursty substructure, these signals are still considered to be a single signal. Conversely, the non-OFDM signals may be bursty or frequency-hopping, where each burst is considered a different signal. The non-OFDM signals are bursty or frequency-hopping with a probability of 0.2 for either behavior. Since non-OFDM signals have smaller bandwidths and a single source may represent numerous bursts, the number of signals of these families are larger. To compensate this imbalance, the probability of signal source selection for OFDM signals is twice as likely to be selected as signals belonging to other modulation families.

When a non-OFDM signal is bursty, its burst duration is randomly selected to be within the range of 0.05 to 0.2 of the full sample duration. Additionally, its silence period is randomly selected to be in the range of 1 to 3 times its burst duration. When the signal is frequency-hopping, its number of frequency channels are randomly selected to be in the range 2 to 16 channels, centered around its original center frequency.

All signals' center frequencies are randomly selected to be in the range -0.4 to 0.4 of the normalized bandwidth of the full example. Bandwidths range from 0.0125 to 0.7 of the full normalized bandwidth of the example; however, as previously stated, OFDM signals bias towards the larger bandwidths. The start and stop samples of each signal source are also randomized. A signal source may start or stop during each example with a probability of 0.2. If a signal source starts at the beginning of the example and stops during the example, the stop point is randomly determined to be in the range 0.05 to 0.95 of the full example duration. Conversely, if a signal source stops at the end of the example and starts during the example, the start point is randomly determined to be in the range 0.0 to 0.95 of the full example duration. Note that all signals in each example are intentionally separated in time and/or frequency such that no signals are overlapping.

OFDM Realism. In the narrowband Sig53 dataset, OFDM signals are generated with randomized sidelobe suppression methods and randomized inclusion or omission of the DC subcarrier. In the WBSig53 dataset, we also introduce additional randomized time and frequency effects with bursty symbols, pilot carriers, and emulated resource blocks. These effects emulate more realistic OFDM communications signals. Figure 7 shows several spectrograms of randomized OFDM signals, where we can see the newly introduced effects appearing randomly throughout the data examples.

Figure 7: OFDM Randomizations

Time Shift Impairment. The time shift impairment applies a randomized shift in time and pads the data back to the desired IQ sample length. Figure 8 shows this impairment's effects when followed by adding noise to account for the zero-padding. Note that the labels are adjusted accordingly.

(a) Original Data   (b) Impaired Data

Figure 8: Time Shift Impairment

Frequency Shift Impairment. The frequency shift impairment applies a randomized shift in frequency. This transform inspects the labels prior to performing the frequency shift in order to ensure aliasing is properly avoided. If a signal is to be shifted past the boundaries, the impairment upsamples the input data, applies the frequency shift, applies a filter, and then downsamples the data to original rate. The labels are adjusted to follow the signal's position post-shift. Figure 9 shows this impairment's effects. Note that the labels are adjusted accordingly and no aliasing occurs with signals at the edges.

(a) Original Data   (b) Impaired Data

Figure 9: Frequency Shift Impairment

Random Resample Impairment. The random resample impairment applies a randomized resampling to the input data. Like the frequency shift impairment, this transform inspects the input labels prior to performing the resampling in order to properly handle/avoid aliasing. Once again, the labels are adjusted to appropriately represent the impaired data's signals. If downsampling occurs, the input data is zero-padded back to the original data size. If upsampling occurs, the input data is truncated back to the original data size. Figure 10 shows the random resample impairment effects.

(a) Original Data   (b) Impaired Data

Figure 10: Random Resample Impairment

Spectral Inversion Impairment. The spectral inversion impairment inverts the frequency components of the input data by multiplying the imaginary values by -1. Once again, the labels are adjusted to track the inverted signals' locations throughout the data. Figure 11 shows the spectral inversion effects.

(a) Original Data   (b) Impaired Data

Figure 11: Spectral Inversion Impairment

Additive White Gaussian Noise Impairment. The additive white Gaussian noise (AWGN) impairment adds noise to the full input data. The magnitude of the AWGN is parameterized to pull from a specified random distribution. Figure 12 shows varying degrees of AWGN applied to the input data.

(a) Original Data   (b) Impaired Data

Figure 12: Additive White Gaussian Noise Impairment

RandAugment Impairment. In addition to the above impairments, we emulate the RandAugment transform for applying additional impairments (Cubuk et al., 2020). RandAugment is given a list of transforms and selects a subset of them to apply to the input data. For WBSig53, we pass in the impairments: magnitude rescaling, RF roll-off, random convolve, Rayleigh fading, drop samples, phase shift, and IQ imbalance (all detailed below). We set RandAugment to randomly select two of these impairments to apply at any given time. Additionally, we condition the magnitude rescaling impairment with a random application rate of 50% even when RandAugment selects it.

Magnitude Rescaling Impairment. The magnitude rescaling impairment randomly selects a time in the input data and then rescales the magnitude by a random rate. This effect emulates an RF front end amplifier adjustment and can be seen in Figure 13.

(a) Original Data (b) Impaired Data

Figure 13: Magnitude Rescaling Impairment

RF Roll-Off Impairment. The RF roll-off impairment randomly selects filter parameters in order to impose a roll-off effect at either the low frequency edge, the high frequency edge, or both. This effect emulates an RF front end filter and can be seen in Figure 14.

(a) Original Data (b) Impaired Data

Figure 14: RF Roll-Off Impairment

Random Convolve Impairment. The random convolve impairment inputs a random distribution of the number of taps in a filter, where each tap is assigned random values in range 0 to 1. The randomly generated filter is then convolved with the input IQ data. An alpha value is also input to the transform, and it is used to dampen the effect of the randomly filtered data by weighting the newly filtered data and inversely weighting the original data and then summing the results. This random convolution is a relatively cheap form of applying a frequency-selective fading model (Figure 15).

(a) Original Data　　　　　　　(b) Impaired Data

Figure 15: Random Convolve Impairment

Rayleigh Fading Impairment. A Rayleigh fading channel is modeled as a finite impulse response (FIR) filter with Gaussian distributed taps. The FIR filter is randomly generated under constraints and then convolved with the input data, simulating the effects of a frequency-selective, time-invariant Rayleigh fading channel. Figure 16 shows this Rayleigh fading channel applied to the full wideband input data; however, the transform could also be applied directly to signal sources if desired.

(a) Original Data　　　　　　　(b) Impaired Data

Figure 16: Rayleigh Fading Impairment

Drop Samples Impairment. The drop samples augmentation randomly drops IQ samples from the input data using randomized values for the drop rate, the size of each dropped region, and the fill methods for how to replace the regions with dropped samples. The fill methods can be statically or randomly set to choose the following methods: front fill, back fill, mean, or zero; where front fill replaces each drop regions' samples with the last previous valid value, back fill replaces each drop regions' samples with the next valid value, mean replaces each drop regions' samples with the mean value of the full data example, and zero replaces each drop regions' samples with zeros. Note that the abrupt discontinuities cause high frequency spectral effects, as see in Figure 17. This transform is loosely based on TSAug's DropOut transform (Wen, 2019).

(a) Original Data   (b) Impaired Data

Figure 17: Drop Samples Impairment

Phase Shift Impairment. Phase shifts are applied by imposing a randomized rotation of IQ pairs about the complex origin. Phase shift effects are best visualized with an IQ constellation plot; however, for consistency, their spectral effects are shown in Figure 18.
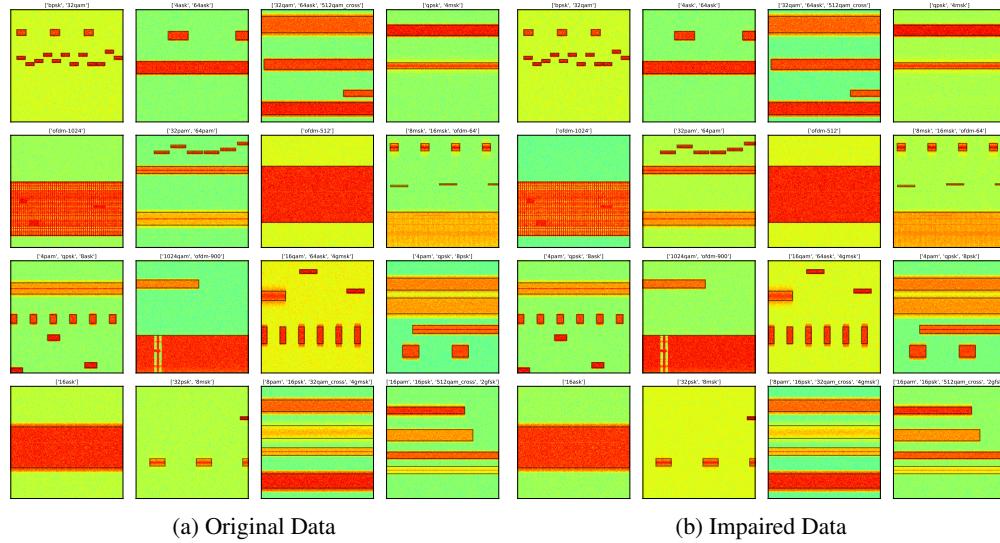

(a) Original Data   (b) Impaired Data

Figure 18: Phase Shift Impairment

IQ Imbalance Impairment. IQ imbalance is applied with three randomized parameters: amplitude imbalance, phase imbalance, and DC offset. Amplitude imbalance causes a growth or reduction of IQ pairs in a constellation, phase imbalance causes a rotation of IQ pairs in a constellation, and DC offset causes a shift in all IQ pairs. These effects are best visualized with an IQ constellation plot; however, for consistency, their spectral effects are shown in Figure 19.

(a) Original Data (b) Impaired Data

Figure 19: IQ Imbalance Impairment

A.2   TOOLS APPENDIX

In addition to the data impairments described in Appendix A.1, the TorchSig toolkit can also apply a number of augmentations to the WBSig53 data or any other dataset. Here, we step through TorchSig's additional data augmentations and show their effects on wideband data. We also extend TorchSig's list of augmentations to include augmentations that can be applied directly to the spectrogram data after the IQ data is transformed.

A.2.1   DATA AUGMENTATIONS

In addition to the impairments used in the generation of the WBSig53 dataset, TorchSig provides support for other signal domain-specific data augmentations available for use during training.

Time Reversal. The time reversal augmentation reverses the order of the IQ samples in the input. Since time reversal in the signal domain also results in a spectral inversion, the TorchSig time reversal augmentation additionally has the option to undo the spectral inversion effect if desired. The time reversal augmentation is shown in Figure 20. Note that the labels are also updated with the augmentation to properly track the updated time and frequency information.

(a) Original Data   (b) Impaired Data

Figure 20: Time Reversal Augmentation

Channel Swap. The channel swap augmentation switches the real and imaginary componets of the input complex data. In the signal domain, this has the same effect as a spectral inversion followed by a static $\pi/2$ phase shift (Figure 21).

(a) Original Data   (b) Impaired Data

Figure 21: Channel Swap Augmentation

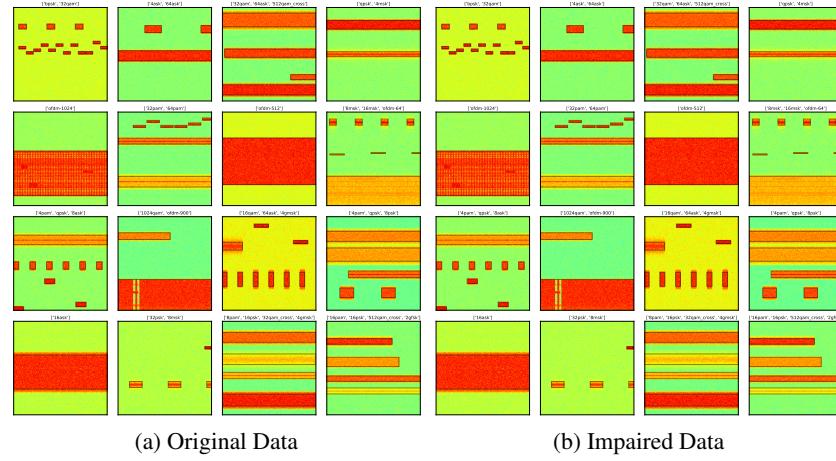**Amplitude Reversal.** Amplitude reversal augments the input data by simply multiplying by -1 (Figure 22).

(a) Original Data   (b) Impaired Data

Figure 22: Amplitude Reversal Augmentation

Quantize. Input data is quantized to a randomly selected number of levels with the quantize data transform, loosely emulating the bit-depth in an analog-to-digital converter (ADC) seen in digital RF systems. The quantization transform also allows for various rounding types between: flooring the observed values to the next-lowest valid quantized value, setting every value in a region to the middle value of the region, or rounding each value to the next largest valid quantized value (Figure 23).

(a) Original Data   (b) Impaired Data

Figure 23: Quantize Augmentation

CutOut. The CutOut transform is a modified version of the computer vision domain's CutOut as seen in DeVries & Taylor (2017). Our version of CutOut inputs randomized cut durations and cut types to select how large a region in time should be cut out. The cut out region is then filled with either zeros, ones, low-SNR noise, average-SNR noise, or high-SNR noise. In the context of wideband data, the CutOut augmentation appropriately omits the cut-out regions of signals. If the cut-out region falls in the middle of a longer signal, the signal label is divided into separate signals of appropriate size. The CutOut effects can be seen in Figure 24.

(a) Original Data   (b) Impaired Data

Figure 24: CutOut Augmentation

PatchShuffle. The PatchShuffle transform is a modified version of the computer vision domain's PatchShuffle as seen in Kang et al. (2017). Our version of PatchShuffle operates solely in the time domain, randomly shuffling multiple local regions of IQ samples, using a randomized patch size input distribution and a randomized shuffle ratio to discern how many of the patches should undergo random local shuffling (Figure 25).
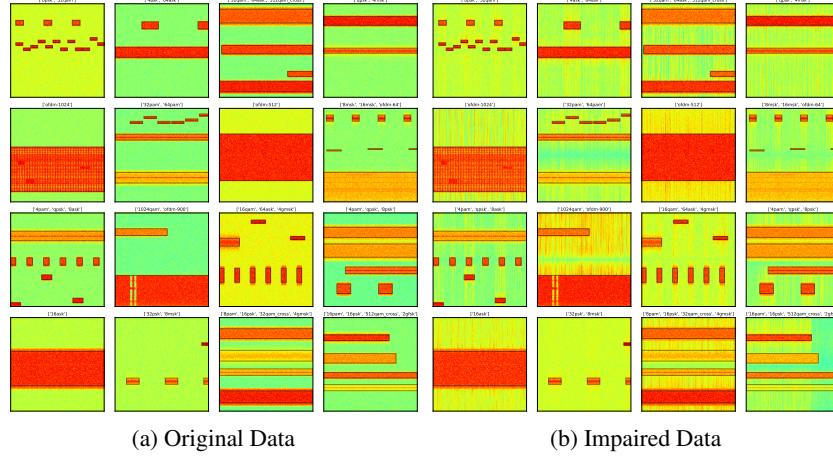

(a) Original Data   (b) Impaired Data

Figure 25: PatchShuffle Augmentation

Local Oscillator Drift. The local oscillator (LO) drift transform emulates the imperfections of a receiver's LO. This transform models the LO drift by implementing a random walk in frequency with a drift rate and a max drift set as inputs, where when the max drift is reached, the frequency offset is reset to 0 (Figure 26).

(a) Original Data   (b) Impaired Data

Figure 26: Local Oscillator Drift Augmentation

Time-Varying Noise. The time-varying transform adds AWGN within a specified low to high SNR range with a specified number of inflection points at which point(s) the slope of the time-varying noise reverses direction (Figure 27).
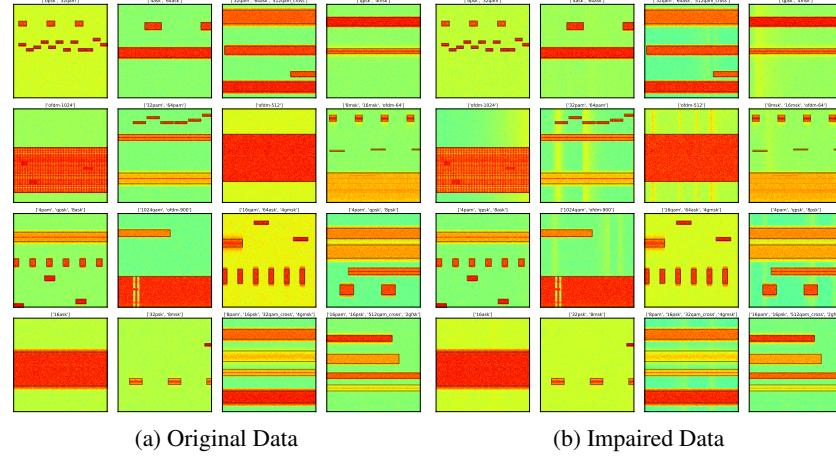

(a) Original Data   (b) Impaired Data

Figure 27: Time-Varying Noise Augmentation

Clip. The clip transform inputs a percentage that it uses to calculate the max and min values allowable through the clipping transform, setting all values above and below these values to the max and min, respectively (Figure 28).

(a) Original Data   (b) Impaired Data

Figure 28: Clip Augmentation

Add Slope. The add slope transform computes the slope between every IQ sample in the input with its preceding sample and adds the slope to its current IQ sample. This transform has the effect of amplifying higher frequency components more than the lower frequency components (Figure 29).
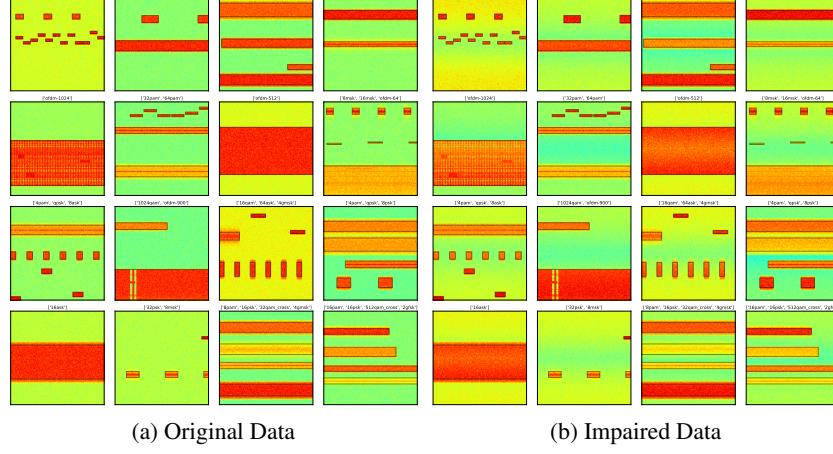

(a) Original Data   (b) Impaired Data

Figure 29: Add Slope Augmentation

Gain Drift. A gain drift transform is also implemented to complement the LO drift transform's frequency effects with magnitude effects. The gain drift transform inputs similar max/min drift values and a drift rate, which are used in a random walk of adjusting the magnitudes of the input data IQ samples over time (Figure 30).

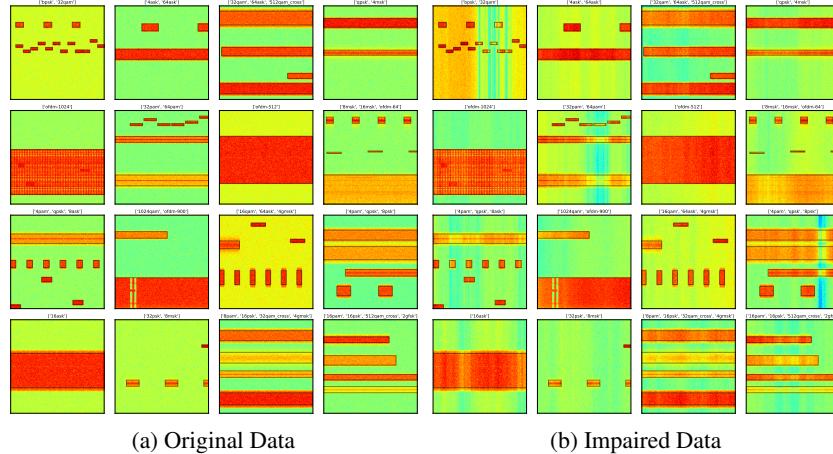(a) Original Data   (b) Impaired Data

Figure 30: Gain Drift Augmentation

Automatic Gain Control. An automatic gain control (AGC) implementation is included in TorchSig as a data transform with an input scaling of default values for randomization across augmentation calls. The default values under random scaling can also be updated with arguments including: an initial gain value, an alpha for averaging the measure signal level, an alpha amount by which to adjust gain when in the tracking state, an alpha value by which to adjust gain when in the overflow state, an alpha value by which to adjust gain when in the acquire state, a reference level specifying the level of intended gain adjustment, a tracking range of allowable deviation before going into the acquire state, a low level which specifies when the AGC is disabled, and a high level which specifies when the AGC enters the overflow state (Figure 31).

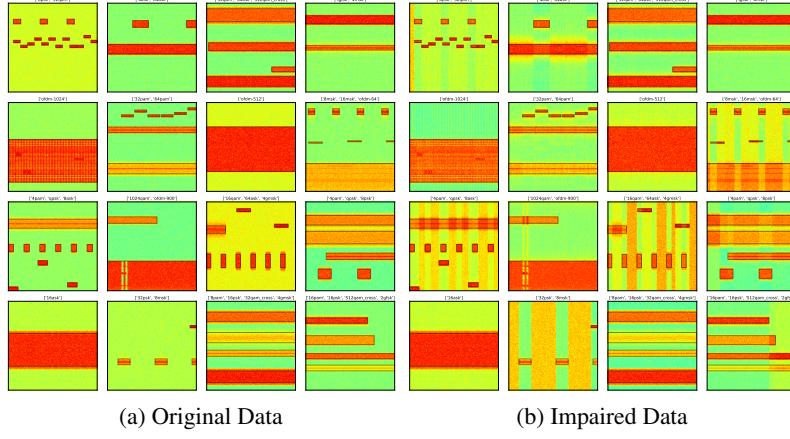(a) Original Data　　　　(b) Impaired Data

Figure 31: Automatic Gain Control Augmentation

Spectrogram Random Resize Crop. A common augmentation used in the vision domain is to randomly resize an input image and then crop the new image down to the expected input size. Here, we emulate this technique by randomizing the parameters of the spectrogram operation. The randomized paramters of the spectrogram operation result in variable sized spectrograms which we then crop down to the expected input size. If the spectrogram is smaller than a desired dimension, we pad with emulated noise (Figure 32).
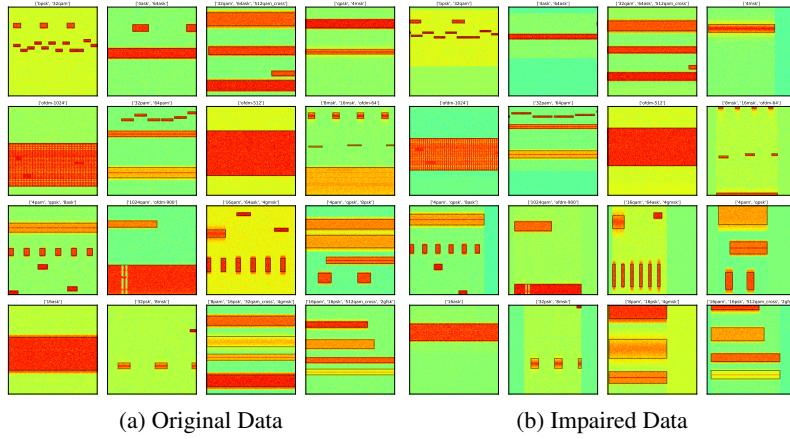

(a) Original Data　　　　(b) Impaired Data

Figure 32: Spectrogram Random Resize Crop Augmentation

MixUp. The wideband MixUp transform implements a modified version of the vision domain's MixUp (Zhang et al., 2018). Our MixUp transform inputs a secondary dataset from which additional data samples are drawn. The new data sample is then mixed with the original data sample, resulting in additional signals being added to the data sample. All labels from the new sample are added to the original label, such that a network can now be tasked with discovering additional, potentially-overlapping signals. This effect is seen in Figure 33.
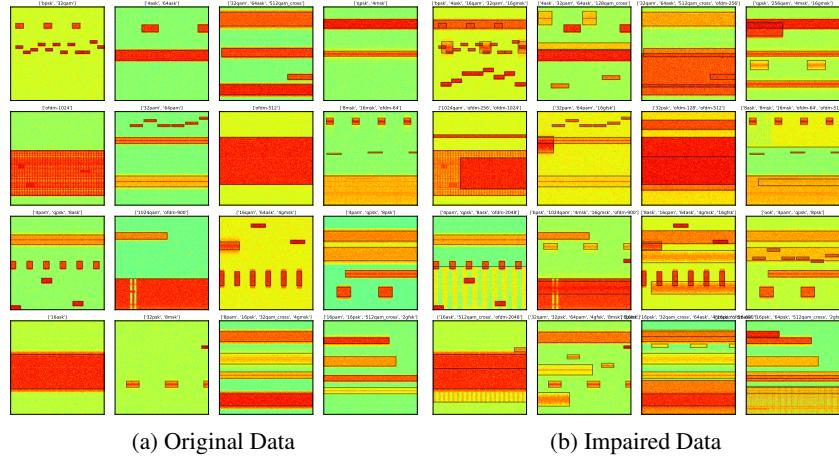

(a) Original Data   (b) Impaired Data

Figure 33: MixUp Augmentation

CutMix. The wideband CutMix transform implements a modified version of the vision domain's CutMix (Yun et al., 2019). Like the MixUp transform, out CutMix transform inputs a secondary dataset from which additional data samples are drawn. A random region of the original data sample is cut out and replaced with a region of appropriate size from the secondary data sample. All labels are adjusted to appropriately capture all old and new signals as seen in Figure 34.
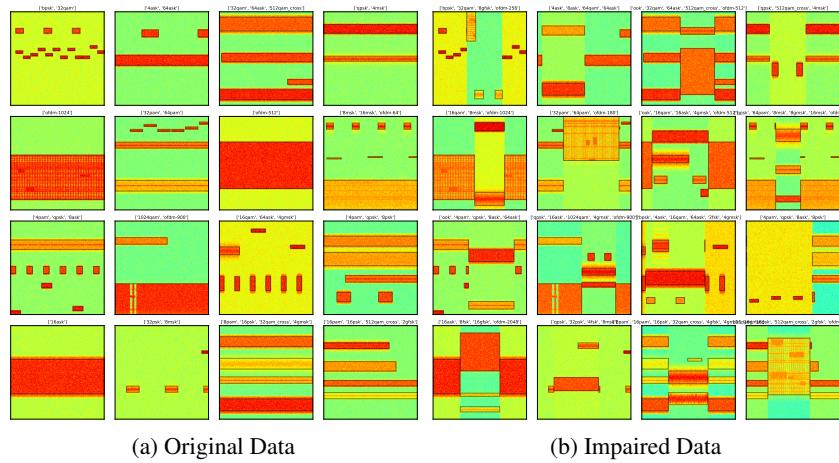

(a) Original Data   (b) Impaired Data

Figure 34: CutMix Augmentation

A.2.2   SPECTROGRAM DATA AUGMENTATIONS

TorchSig also now supports data augmentations that are applied after the raw IQ data has been converted to a spectrogram representation. Details on these spectrogram data augmentations are below.

Spectrogram Resize. The spectrogram resize transform inputs a spectrogram and resizes the data and labels to the specified dimensions. If the input is larger than the desired size, the input is cropped. If the input is too small to meet the desired size, the outer regions are padded with emulated noise as seen in Figure 35.

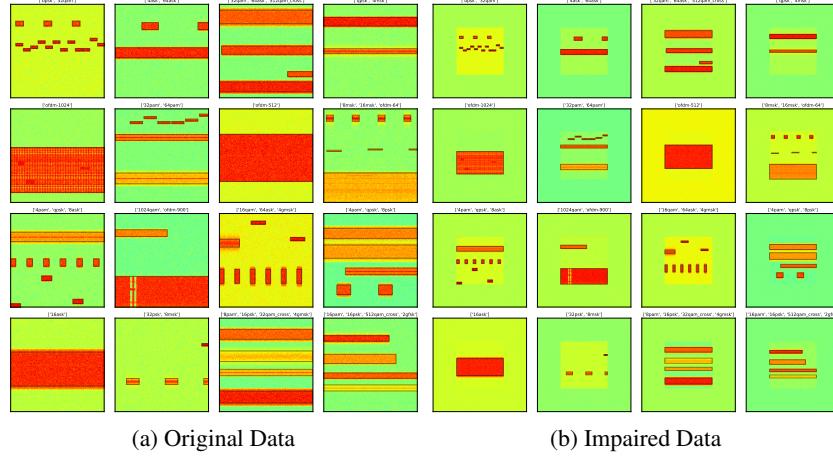(a) Original Data   (b) Impaired Data

Figure 35: Spectrogram Resize Transform

Spectrogram Drop Samples. The spectrogram drop samples transform mirrors the IQ data's drop sample transform by randomly dropping samples throughout an input spectrogram. Similar to the IQ data drop samples transform, the dropped samples are replaced by either the front fill, back fill, the mean value, zeros, low values, the minimum value, the maximum value, and/or ones, depending on the input options provided. This effect can be seen in Figure 36.

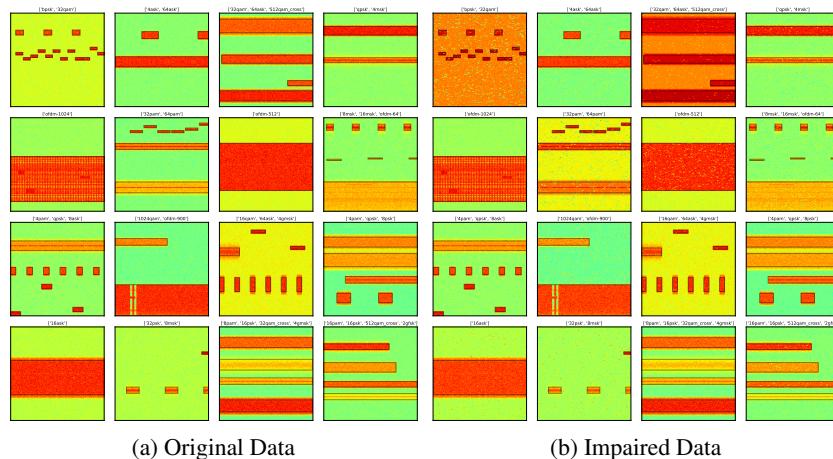(a) Original Data   (b) Impaired Data

Figure 36: Spectrogram Drop Samples Augmentation

Spectrogram PatchShuffle. The spectrogram PatchShuffle transform mirrors the IQ data's PatchShuffle transform, but instead of shuffling local regions in time only, the spectrogram implementation performs the shuffling in time and frequency rectangular regions. This effect is most easily seen at the edges of signals and can be viewed in Figure 37.
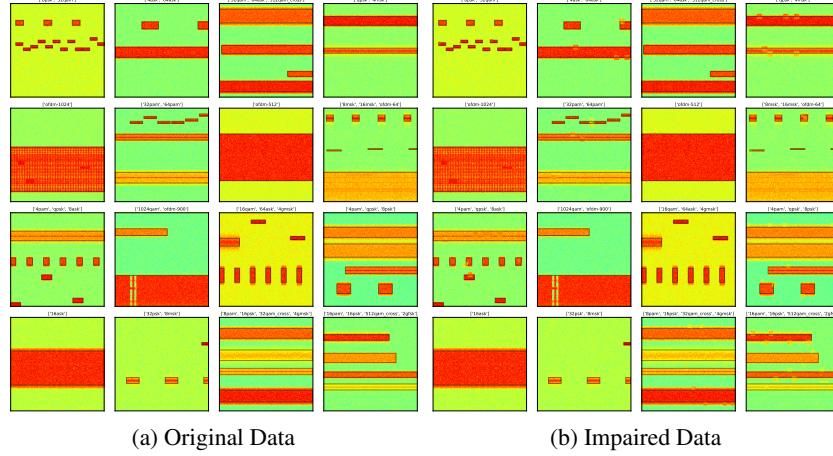

(a) Original Data (b) Impaired Data

Figure 37: Spectrogram PatchShuffle Augmentation

Spectrogram Translation. The spectrogram translation augmentation applies a time and frequency shift of the spectrogram and fills new regions with emulated background noise (Figure 38).
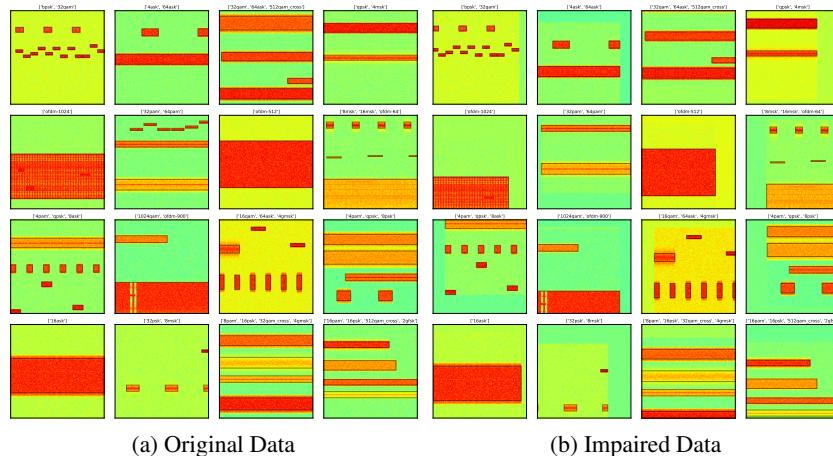

(a) Original Data (b) Impaired Data

Figure 38: Spectrogram Translation Augmentation

Spectrogram Mosaic Crop. The spectrogram mosaic crop transform inputs a secondary dataset that is also transformed to an equivalent spectrogram representation, randomly samples three additional examples from the secondary dataset, forms a $2x2$ grid of the four examples, and then randomly crops a region the size of a single spectrogram. The resulting spectrogram consists of signals from up to all four of the examples used in the generation of the grid. Figure 39 shows examples of this augmentation.
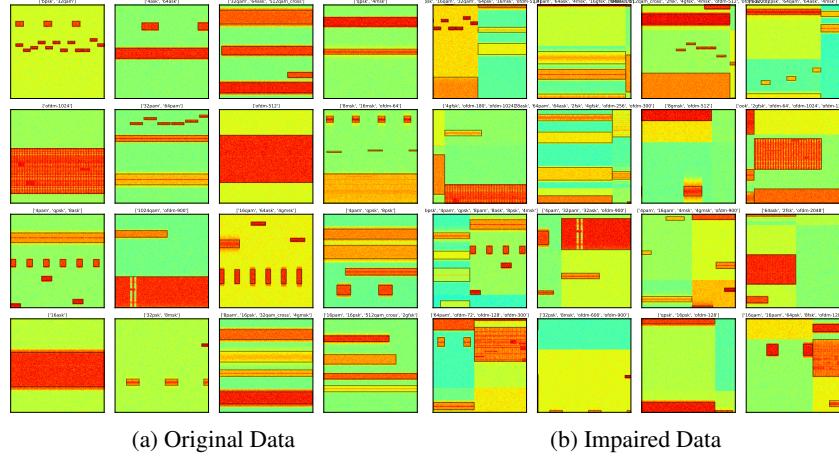

(a) Original Data   (b) Impaired Data

Figure 39: Spectrogram Mosaic Crop Augmentation

Spectrogram Mosaic Downsample. The spectrogram mosaic downsample transform is similar to the spectrogram mosiac crop transform; however, instead of cropping out a region, it downsamples the input such that all four spectrograms are fully contained at a lower resolution. Note that this effect results in many smaller signals present in the transformed data. Figure 40 shows examples of this augmentation. Note that the impaired data does retain all of the label information for each signal, but in the figure, the labels are omitted so the smaller signals can be more easily seen.
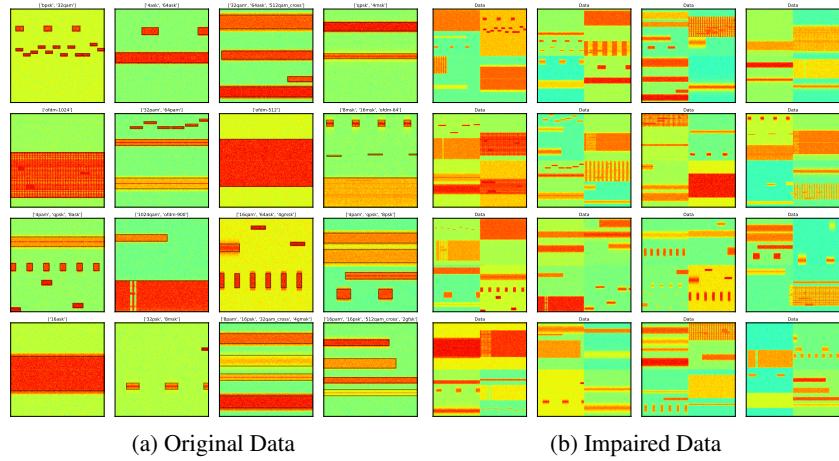

(a) Original Data   (b) Impaired Data

Figure 40: Spectrogram Mosaic Downsample Augmentation

A.3   NETWORK BACKGROUNDS

A.3.1   YOU ONLY LOOK ONCE (YOLO)

Prior to the introduction of YOLO, two stage detection methods detected possible object regions in an input image during the first stage and then (after feeding this information to another subsystem) classified the image in each of these regions during the second stage. In Redmon et al. (2015), a new approach is introduced in which the neural network make bounding box predictions and class probabilities all at once. This algorithm produces SoTA results at much faster speeds.

The YOLOv1 algorithm breaks the input image into an array of $S \times S$ grid cells. Each grid cell produces B bounding boxes and C conditional class probabilities, where C is the number of possible classes. Each bounding box contains 5 predictions, representing its (x, y) center location, width, height, and confidence score. This confidence score incorporates the probability that the bounding box contains an object as well as the accuracy of the bounding box location and size. Thus, the predictions produced by each input image are encoded as an output tensor of size $S \times S \times (B \times 5 + C)$. The system proposed in Redmon et al. (2015) assigns $S = 7$, $B = 2$ while processing images with $C = 20$ classes. Thus, the output tensor has size: $7 \times 7 \times 30$. Non-maximal suppression is used to pare down the information from this output tensor to a final set of object classes and their respective locations. Note that every grid cell predicts only a single set of C conditional class probabilities. Thus, objects of only one class are predicted from each grid cell. What if two different classes of objects have centers that fall into the same grid cell? This is addressed in YOLOv2.

YOLOv2 is introduced in Redmon & Farhadi (2016), which has many improvements over YOLOv1. Changes are made to the network as well as the training procedures. A notable change is the replacement of bounding boxes with anchor boxes. The authors choose to break each input image into an array of 13×13 grid cells, with each grid cell containing 5 anchor boxes. The authors also introduced a method that makes use of the training data to determine the width and height of the anchor boxes. Another notable change in YOLOv2 is that it can handle input images of different sizes. The same authors published Redmon & Farhadi (2018) a couple of years later. Additional incremental improvements are made to create YOLOv3. More improvements are made to the YOLO algorithm in Bochkovskiy et al. (2020) and Jocher et al. (2022), creating YOLOv4 and YOLOv5 respectivey.

This effort uses YOLOv5 of various sizes obtained from the original authors' GitHub repository. These model are (from smallest to largest): YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. As these YOLO models get larger, the performance improves and the speed decreases. We create a sixth model, called YOLOv5p (the "pico" model) which is smaller than the YOLOv5n model and expected to run much faster. For this work, we explore the signal detection and signal recognition performance of YOLOv5p, YOLOv5n, and YOLOv5s on the WBSig53 impaired dataset.

A.3.2   DETECTION TRANSFORMER (DETR)

The DEtection TRansformer (DETR) was introduced in Carion et al. (2020) as a direct set prediction technique that effectively removes the hand-crafted anchors and non-maximal suppression procedures of previous object detection approaches. DETR accomplishes its goal of streamlining the detection pipeline by employing a convolutional backbone that feeds a transformer encoder-decoder architecture and by forcing unique predictions via a bipartite matching set-based global loss function. The convolutional backbone reduces the input dimensionality to a learned 2D representation. The learned representation is then flattened, supplemented with positional encodings, and then passed as input to the transformer encoder. The transformer decoder receives the encoded representation and inputs N learnable object queries, where N represents a number larger than the maximum number of objects expected in any input. The output of the transformer decoder is N learned embeddings that are passed to N multi-layered perceptron (MLP) prediction heads. These prediction heads infer the class and bounding box of unique objects present within the input.

During training, the outputs of the prediction heads are compared to the target labels by using a Hungarian matching loss function Kuhn & Yaw (1955) that jointly measures a linear combination of the class loss and localization loss. The Hungarian matching function forces unique pairs of predicted objects with target objects, where inputs with fewer than N objects are padded to N with ϕ (no object). The class loss is a negative log-likelihood score between class prediction and targets. The

localization loss is a weighted sum of the commonly-used l_1 loss with a scale-invariant generalized IoU loss Rezatofighi et al. (2019).

In the original work, the DETR authors experimented with ResNet-50 and ResNet-101 backbones He et al. (2015) with a vanilla transformer Vaswani et al. (2017). In our work, we adapt this architecture slightly by using EfficientNet backbones Tan & Le (2019) with XCiT transformers El-Nouby et al. (2021). By making these slight modifications, our networks are significantly more data efficient and run faster, primarily due to the XCiT transformer's linear complexity with respect to the input length. Similar to YOLOv5, we experiment with three scales of this architecture. We experiment with the following backbones: EfficientNet-B0, EfficientNet-B2, and EfficientNet-B4. For all architectures, we leave the transformer constant with the XCiT-Nano scale. We refer to these DETR architectures as: DETR-B0-Nano, DETR-B2-Nano, and DETR-B4-Nano.

A.3.3 PYRAMID SCHEME PARSING NETWORK (PSPNET)

The Pyramid Scene Parsing Network (PSPNet) is a fully convolutional neural network for semantic segmentation tasks. PSPNet consists of an encoder that compresses the input into a learned feature map and a decoder that is built of a pyramid pooling module. The pyramid pooling module inspects and fuses features under four different scales before upsampling and concatenating to form the final feature representation, which contains local and global contextual information. As a final step, the representation is fed into a convolutional layer that outputs the pixel-by-pixel class predictions.

In the original work, the authors experiment with various scale ResNet encoders. In our work, we adapt this architecture slightly by replacing the ResNet encoders with EfficientNet encoders. Once more, we conduct our experiments using multiple scales of encoders with: EfficientNet-B0, EfficientNet-B2, and EfficientNet-B4. We refer to these networks as: PSPNet-B0, PSPNet-B2, and PSPNet-B4, respectively.

A.3.4 MASK2FORMER

Mask2Former: The Masked-attention Mask Transformer (Mask2Former) is an architecture capable of performing semantic, instance, or panoptic segmentation. At the time of its release, Mask2Former set SoTA performance across all three segmentation categories, outperforming models specialized in each category. Mask2Former builds on the architecture introduced in MaskFormer Cheng et al. (2021b). This architecture consists of a backbone, a pixel decoder, and a transformer decoder. Mask2Former improves upon the original MaskFormer architecture by introducing masked attention in the transformer decoder. The masked attention modulates the standard cross-attention matrix by imposing a binarized mask of the previous transformer decoder layer's mask predictions to the current key-query pairs. The effect of this masking is that the attention only occurs within the foreground region of the predicted mask for each query, rather than the full feature map. Mask2Former also uses high-resolution features from the pixel decoder at multi-scales by feedings one scale to differing transformer decoder layers one at a time. Additionally, Mask2Former inverts the order of the self and masked attention, makes the transformer decoder's queries learnable, and removes dropout for computational effectiveness. All of these improvements over MaskFormer results in a flexible neural network that trains significantly faster with higher performance.

In the original work, the authors conduct experiments with ResNet and Swin-L Liu et al. (2021) backbones pretrained on ImageNet-22K Russakovsky et al. (2015). The pixel decoder in their experiments is a multi-scale deformable attention transformer (MSDeformAttn) Zhu et al. (2020). The transformer decoder is a vanilla transformer decoder with 9 layers, 100 queries, and auxiliary losses at each layer. The loss function is a weighted combination of a classification loss and a mask loss, where the mask loss consists of a binary cross-entropy loss and a dice loss Milletari et al. (2016).

In our work, we explore the Mask2Former as implemented by the original authors; however, we modify the backbone. To better mirror our other networks' backbones/encoders, we use various scale EfficientNets for the Mask2Former backbone. Once again, we use EfficientNet-B0, EfficientNet-B2, and EfficientNet-B4, and we refer to these networks as Mask2Former-B0, Mask2Former-B2, and Mask2Former-B4, respectively. Unlike the original authors, our backbones are not pretrained on any task prior to their use within the full Mask2Former architecture.