ABSTRACT

Traditionally source identification is solved using threshold based energy detection algorithms. These algorithms frequently sum up the activity in regions, and consider regions above a specific activity threshold to be sources. While these algorithms work for the majority of cases, they often fail to detect signals that occupy small frequency bands, fail to distinguish sources with overlapping frequency bands, and cannot detect any signals under a specified signal to noise ratio. Through the conversion of raw signal data to spectrogram, source identification can be framed as an object detection problem. By leveraging modern advancements in deep learning based object detection, we propose a system that manages to alleviate the failure cases encountered when using traditional source identification algorithms. Our contributions include framing source identification as an object detection problem, the publication of a spectrogram object detection dataset, and evaluation of the RetinaNet and YOLOv5 object detection models trained on the dataset. Our final models achieve Mean Average Precisions of up to 0.906. With such a high Mean Average Precision, these models are sufficiently robust for use in real world applications.

1. INTRODUCTION

Since the inception of YOLO[10], the object detection space has been dominated by deep learning based architectures. With the popularization of deep learning based object detection approaches, object detection models have become accuracy, robust and reliable.

By reframing source identification as an object detection problem, we translate these advancements in deep learning based object detection to the source detection problem. Our system manages to tackle the failures cases of traditional activity-threshold based systems. These failure cases include failing to distinguish distinct sources, failure to detect signals occupying small frequency bands, and failure to detect sources below the signal to noise floor.

Alongside our publication we open source everything required to reproduce our results. This includes our annotated

Our fully open source codebase is available at [https://github.com/lukewood/em-loader](https://github.com/lukewood/em-loader)

Fig. 1. A sampling of images and detections using the YOLOv5 (top) RetinaNet (bottom) model
spectrogram dataset, training pipeline, and inference pipeline. Our final models include both YOLOv5 and RetinaNet models, capable of scoring up to a 0.906 Mean Average Precision on our dataset.

Our contributions include the framing of source identification as an object detection problem, a novel object detection dataset consisting of spectrograms and bounding box annotations to accompany, a deep learning based source identification system trained on the novel dataset, analysis of the distinctions between traditional object detection datasets and our dataset, and finally a comparison between our deep learning based approach and traditional energy based systems. Our contributions can be broken into three components: problem framing, dataset production, and model training.

We present a novel object detection dataset as well as propose solutions to the dataset including a trained RetinaNet[7] and YOLOv5[6] architecture.

Datasets of similar format include Common Objects in Context[8], Pascal VOC[1], and Kitti[3]. Our dataset differs primarily in the fact that instead of natural images, it consists of spectrogram images.

Other object detection architectures that could be used to solve our dataset include Faster-RCNN[2], DINO SWIN-L[9], YoloX[2], and Yolov3[11].

We evaluate our models using MSCOCO metrics. These metrics were originally explained in the Common Objects in Context publication[8]. To evaluate our MSCOCO metrics we use both the pycocotools and KerasCV metric implementations[14].

2. DATA GENERATION

To demonstrate the effectiveness of our methodology, we present a synthetic dataset generated by Matlab’s Communications Toolbox. Our dataset consists of 4686 annotated spectrogram samples. These samples contain a 512x512 image and 0-13 sources, each with a corresponding bounding box annotation.

The samples in the dataset consist of a spectrogram containing a variety of signals and annotations of the signal bounding boxes. All spectrograms were constructed from I/Q samples that were generated using the Matlab communications and signal processing toolboxes with a sampling rate of 100 MS/s and a total transmission time of 50 ms. The signal types appearing in this dataset are DSSS, BLE, QAM, WiFi, AM, and FM, with each sample having between one and four signals from this set. Signal metadata like center frequency, bandwidth, arrival time, and signal-to-noise ratio (SNR) are uniformly randomized between samples.

To generate samples for the dataset, a combination of signal types is first selected. Next, we initialize the signal source and add white Gaussian noise. The center frequency, bandwidth, and SNR for each signal type are randomly selected, a signal generator is instantiated, and each signal is added to the source. After this signal metadata is configured, I/Q samples for 5 realizations of this configuration are generated with signal durations and arrival times being randomized between realizations.

We repeat this process of randomly selecting signal metadata and generating realizations 20 times for a given combination of signals, resulting in 100 total realizations of signal-level metadata for every combination. After all realizations are generated, the source is cleared, the I/Q samples and metadata are stored, and the entire process is repeated for a new combination of signals.

Spectrograms are constructed from the I/Q samples using a sampling rate of 100 MS/s, 1024 discrete Fourier transforms, an overlap length of 128 samples, and a Hanning window of length 256. The spectrograms are resized to 512x512 with bicubic interpolation and saved as PNG images. The coordinates for the bounding box annotations for the signals in an image are calculated from the signal metadata and saved in a corresponding text file. These images and labels are the final dataset used in training our models.

Figure[1] shows a collection of annotated ground truth samples from the generated dataset.

2.1. Difficulties of Spectrogram Object Detection

In our initial experiment, we train a RetinaNet with the library default settings for the RetinaNet. This includes the default settings for IoU threshold for label encoding, IoU threshold for label decoding, and the default configuration for the AnchorBox generation process.

In initial experiments, the loss converged to extremely low values on both the training and validation sets. Despite this, the model only achieves an MaP of 0.205 and a Recall of 0.238. These are intolerable results for use in a production system.

Upon investigation of some sample predictions, it becomes clear that the model is able to detect some source classes with both high precision and recall, while entirely missing other source classes. The medium sized signals with roughly equal side lengths are detected by the detector with low variance between the ground truths and the predicted boxes. The small boxes, those with high aspect ratios, those with low aspect ratios, and large boxes are never detected at all.

The reason that the loss converges to such a low loss but the MaP and Recall remain low stems from the strange shapes of the objects in the spectrograms. While most objects in the natural world follow a normal distribution for aspect ratio due to the influence of gravity, spectrograms have no such property. This leads the sources present in the spectrograms to have a wide array of aspect ratios. As such, the anchor boxes generated by the default configuration used in the KerasCV RetinaNet are unlikely to match with the boxes from our training dataset in the label encoding process. This leads to the
boxes that are not encoded having no representation in the encoded batch of encoded training targets. This explains the low loss, low Recall, and low MaP.

The reason that anchor configuration is particularly important for spectrogram object detection can be attributed to the wide, non-Gaussian distribution of aspect ratios and side lengths of the spatial representations of the bounding box annotations. This is shown in a histogram plot in figures 2 and 3.

This differs greatly from object annotations present in natural images. For comparison, the aspect ratio and side length distributions for PascalVOC can be seen in figures 2 and 3.

3. MODEL TRAINING & EXPERIMENTAL RESULTS

Using the dataset produced using the process described in section 2, we can train any deep learning based object detection model. Our models are trained on a pre-generated train test split. This split consists of 3859 training images, and 827 test images. Our object detection models are evaluated based on two metrics: the standard variant of Mean Average Precision and Recall used in the MSCOCO challenge. We leverage the KerasCV COCO metric implementations, and parameterize them as described in [13]. Using these implementations enables to perform train time evaluation of these metrics.

Due to the high cost of computation required to compute MaP we do not evaluate the true MaP of our model during training. Instead, we approximate our COCO metrics by evaluating them for a subset of 20 of the images from the evaluation dataset. Using this proxy, we can evaluate our model’s performance across epochs and monitor its train time progress. Final metrics are evaluated on the entirety of the test set. As an additional inference test, we manually examine the visual results of the predictions.

3.1. RetinaNet

In the first experiment, RetinaNet is trained using the KerasCV [14] library. The RetinaNet [7] architecture uses a ResNet50 [4] backbone, which can achieve approximately 30 frames per second on a standard consumer GPU. With a MobileNetv3 [5] backbone, a RetinaNet can achieve up to 60 FPS on a consumer GPU, enabling real time source identification. In this experiment, we configure the aspect ratios for the anchor box generator according to the results of section 2.1. We do not configure the anchor generator according to the side lengths. This results in the model still not detecting the small boxes.

During training, spectrograms are loaded into memory as Tensors of shape $(512, 512, 3)$ with raw pixel values in the range of $[0, 255]$. Pixel values are rescaled to the range $[0, 1]$ by simply dividing them by 255.

The backbone used is a ResNet50, with weights initialized using the weights produced by training a ResNet50 to perform image classification. Our feature pyramid, prediction heads, and backbone are all trained using the spectrogram dataset. A SGD optimizer with a global clip norm [15] of 10.0 is used for fitting, with a batch size of 8. Without the global clip norm, the loss explodes due to steep gradients existing at many points in the loss landscape. Training is lightweight; ours being performed on a single GPU A100. The learning rate of our optimizer is reduced after 5 epochs of training with no loss improvement on the validation set, and training stops once no improvement has been seen in 20 epochs.

No data augmentation was used to train the RetinaNet model. While we experimented with data augmentation, we found that it decreased performance instead of improved performance. This is due to the fact that spectrograms have a significantly different data distribution than natural images. As such, traditional augmentation techniques such as RandomFlip, RandomShear, and others are not valid operations.

Losses alongside MaP and Recall metrics are shown in figures 4. Results for all metrics are shown in table 1.

Upon convergence the RetinaNet model achieves a Recall of 0.565 and MaP of 0.492.

3.2. YOLOv5

To test the usefulness of various augmentation schemes and superior anchor configuration, we train a model using the pre-configured YOLOv5 framework maintained by Ultralytics [6]. The YOLOv5s architecture consists of a CSPDarknet53, a PANet feature pyramid, and a YOLOv4 head to generate the final output vectors with bounding boxes, class probabilities, and objectness scores. The Ultralytics YOLOv5 framework automatically tunes anchor boxes for aspect ratio and side
length, and includes the Mosaic augmentation\[6\] technique. The mosaic augmentation is a logical choice for spectrogram object detection as due to the spatial meaning embedded in spectrograms translations, rotations, and color based augmentations all become meaningless.

Once again for training, spectrograms are loaded into memory as Tensors of shape (512,512,3) with normalized pixel values in the range \([0,1]\). A stochastic gradient descent optimizer is used with a batch size of 16 and a warmup scheduler with a relatively low learning rate for 3 epochs as it ramps up to the normal learning rate. Training is done on a single NVIDIA T4 GPU over 50 epochs and set to stop once no improvement has been seen in 20 epochs.

The loss function for YOLOv5 is a summation of a box loss, objectness loss, and classification loss. In our case, the classification loss is ignored as there is only a single class. Losses and COCO metrics are shown in Figure 5. The YOLOv5\[s\] model achieves much better Recall and MaP than RetinaNet.

### 3.3. Experimental Analysis

Our experiments show that a deep learning based approach to source identification can be highly robust, the importance of anchor configuration in spectrogram object detection, and the effectiveness of the Mosaic data augmentation. In training our RetinaNet, we initially did not tune the anchor generator at all. This yielded incredibly poor results, with a final MaP of approximately 0.2. Upon configuring only the aspect ratios, we manage to score an MaP of 0.492. Finally, we train a YOLOv5 model with optimal anchor configuration and achieve a MaP of 0.906.

In addition to the optimal anchor configuration, we determine that the mosaic augmentation significantly boosts performance. This is a reasonable result; given that most other data augmentation techniques are no longer applicable in the spectral domain. For example, rotations, color jitter, and many other common augmentations are no longer suitable when working with spectrograms. Mosaic, on the other hand maintains the spatial structure of most the sources while still producing synthetic data. Final results of our two best models are available in table 1.

### Table 1. Metrics for both the YOLOv5 and RetinaNet models.

| Model   | Mean Average Precision | Recall   |
|---------|------------------------|----------|
| YOLOv5  | 0.906                  | 0.980    |
| RetinaNet | 0.492                | 0.565    |

Our efforts result in a model that manages to achieve a final Mean Average Precision of 0.906 and Recall of 0.980. These metrics indicate that the model detects almost all sources boxes while making minimal false positive predictions. Our model is sufficiently robust for deployment in a real world system.

### 4. CONCLUSION

Through reframing of source identification as an object detection problem we leverage advancements in deep learning based object detection to handle the failure cases of traditional source identification systems. These failure cases include failing to distinguish distinct sources, failure to detect signals occupying small frequency bands, and failure to detect sources below the signal to noise floor.

We present and open source a novel dataset consisting of 7000 spectrograms alongside bounding box annotations of the sources present in these spectrograms, an open source Python library to load the dataset into a TensorFlow dataset, an open source training script to train a KerasCV RetinaNet on our novel dataset, and a sample solution to the dataset using a YOLOv5 model. Alongside our data and code contributions, we present analysis of the data to produce optimal anchor box configuration for deep learning based object detection systems. Our findings show the distinct importance of anchor configuration in spectrogram object detection.

Our efforts result in a model that manages to achieve a final Mean Average Precision of 0.906 and Recall of 0.980.

Through reframing of source identification as an object detection problem we leverage advancements in deep learning based object detection to handle the failure cases of traditional source identification systems. These failure cases include failing to distinguish distinct sources, failure to detect signals occupying small frequency bands, and failure to detect sources below the signal to noise floor.

We present and open source a novel dataset consisting of 7000 spectrograms alongside bounding box annotations of the sources present in these spectrograms, an open source Python library to load the dataset into a TensorFlow dataset, an open source training script to train a KerasCV RetinaNet on our novel dataset, and a sample solution to the dataset using a YOLOv5 model. Alongside our data and code contributions, we present analysis of the data to produce optimal anchor box configuration for deep learning based object detection systems. Our findings show the distinct importance of anchor configuration in spectrogram object detection.

Our efforts result in a model that manages to achieve a final Mean Average Precision of 0.906 and Recall of 0.980. These metrics indicate that the model detects almost all sources boxes while making minimal false positive predictions. Our model is sufficiently robust for deployment in a real world system.
5. REFERENCES

[1] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[2] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.

[5] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3, 2019.

[6] Glenn Jocher, Ayush Chaursia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Kalen Michael, Jiacong Fang, imyhxy, Lorna, Colin Wong, Zeng Yifu, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvK-itDe, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Max Strobel, Mrinal Jain, Lorenzo Mammana, and xylieong. ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations, August 2022.

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[9] 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*, pages 10012–10022, 2021.

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[13] Luke Wood and Francois Chollet. Efficient graph-friendly coco metric computation for train-time model evaluation, 2022.

[14] Luke Wood, Scott Zhu, François Chollet, et al. Kerascv. [https://github.com/keras-team/keras-cv](https://github.com/keras-team/keras-cv) 2022.

[15] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity, 2019.