Publications of the Astronomical Society of the Pacific, 132:114501 (12pp), 2020 November https://doi.org/10.1088/1538-3873/abaaf7
© 2020. The Astronomical Society of the Pacific. All rights reserved. Printed in the U.S.A.

Narrow-band Signal Localization for SETI on Noisy Synthetic Spectrogram Data

Bryan Brzycki[1], Andrew P. V. Siemion[1,2,3,4], Steve Croft[1,2], Daniel Czech[1], David DeBoer[1], Julia DeMarines[1], Jamie Drew[5], Vishal Gajjar[1], Howard Isaacson[1,6], Brian Lacki[7], Matthew Lebofsky[1], David H. E. MacMahon[1], Imke de Pater[1], Danny C. Price[1,8], and S. Pete Worden[5]

[1] Department of Astronomy, University of California Berkeley, Berkeley CA 94720, USA; bbrzycki@berkeley.edu
[2] SETI Institute, Mountain View, CA, USA
[3] Department of Physics and Astronomy, University of Manchester, Manchester, UK
[4] University of Malta, Institute of Space Sciences and Astronomy, Malta, Italy
[5] The Breakthrough Initiatives, NASA Research Park, Bld. 18, Moffett Field, CA, 94035, USA
[6] University of Southern Queensland, Toowoomba, QLD 4350, Australia
[7] Breakthrough Listen, Department of Astronomy, University of California Berkeley, Berkeley CA 94720, USA
[8] Centre for Astrophysics & Supercomputing, Swinburne University of Technology, Hawthorn, VIC 3122, Australia
Received 2020 June 7; accepted 2020 July 30; published 2020 September 21

Abstract

As it stands today, the search for extraterrestrial intelligence is highly dependent on our ability to detect interesting candidate signals, or technosignatures, in radio telescope observations and distinguish these from human radio frequency interference (RFI). Current signal search pipelines look for signals in spectrograms of intensity as a function of time and frequency (which can be thought of as images), but tend to do poorly in identifying multiple signals in a single data frame. This is especially apparent when there are dim signals in the same frame as bright, high signal-to-noise ratio (S/N) signals. In this work, we approach this problem using convolutional neural networks (CNN) as a computationally efficient method for localizing signals in synthetic observations resembling data collected by Breakthrough Listen using the Green Bank Telescope. We generate two synthetic data sets, the first with exactly one signal at various S/N levels and the second with exactly two signals, one of which represents RFI. We find that a residual CNN with strided convolutions and using multiple image normalizations as input outperforms a more basic CNN with max pooling trained on inputs with only one normalization. Training each model on a smaller subset of the training data at higher S/N levels results in a significant increase in model performance, reducing root mean square errors by at least a factor of 3 at an S/N of 25 dB. Although each model produces outliers with significant error, these results demonstrate that using CNNs to analyze signal location is promising, especially in image frames that are crowded with multiple signals.

Key words: Technosignatures – Search for extraterrestrial intelligence – Astrobiology – Convolutional neural networks

Online material: color figures

1. Introduction

Many avenues in the search for extraterrestrial intelligence (SETI) are largely reliant on our ability to pick out interesting signals in a sea of optical and radio telescope data. Since the 1960s, radio searches for evidence of extraterrestrial intelligence (ETI) have increased in scope in tandem with our improving technology, covering larger instantaneous bandwidths and surveying more targets than before (Drake 1961; Werthimer et al. 1985; Horowitz et al. 1986; Korpela et al. 2001; Welch et al. 2009; Siemion et al. 2013; Wright et al. 2014; MacMahon et al. 2018; Price et al. 2018).

The Breakthrough Listen (BL) initiative is the most thorough SETI search effort, with access to top radio telescopes across the world specifically for use in SETI searches, including 20% of the telescope time on the Green Bank Telescope (GBT) in West Virginia, USA and 25% time on the CSIRO Parkes radio telescope in New South Wales, Australia (Isaacson et al. 2017; Worden et al. 2017; MacMahon et al. 2018; Price et al. 2018). In optical wavelengths, the search uses the Automated Planet Finder at the Lick Observatory in California, USA (Vogt et al. 2014). The BL search has expanded to include such facilities as the MeerKAT telescope in South Africa (Jonas 2009), the VERITAS Cerenkov Telescope at the Whipple Observatory in Arizona, USA (Weekes et al. 2002), the Murchison Widefield Array in Western Australia (Tingay et al. 2018), and the FAST telescope in Guizhou Province, China (Zhang et al. 2020). Sifting through the sheer data volume collected, which can be on the order of hundreds of terabytes per day, is computationally expensive alone, but identifying interesting, anomalous signals is itself a tough open problem.

Most of the coherent radio signals that we observe in BL data are anthropogenic, termed radio frequency interference (RFI). Types of RFI include satellite telemetry, cellular mobile broadcasts, wireless internet, and a host of other artificial sources. These are all types of narrow-band signals, which means each signal has a small frequency bandwidth (generally of order less than 1 kHz). On the other hand, natural astrophysical phenomena usually produce broad-band signals. The challenge for technosignature searches is that if an intelligent civilization is producing signals at radio frequencies (technosignatures), either as directed transmissions or as by-products of advanced technology, these signals are also likely to be narrow-band and therefore appear similar to RFI. SETI searches to date have found mountains of RFI signals, but no conclusive evidence of technosignatures (Tarter 2001; Korpela et al. 2011; Siemion et al. 2013, 2015; Harp et al. 2016; Enriquez et al. 2017; Gray & Mooley 2017; Tingay et al. 2018; Wright et al. 2018; Price et al. 2019).

The science data we collect from radio telescopes are generally stored as arrays of detected intensity (Stokes-*I*) as a function of time and frequency. These can be visualized as dynamic spectra or waterfall plots, with frequency on the *x*-axis, time on *y*-axis, and intensity as a color according to a colorscale. In other words, each pixel corresponds to an intensity value computed at that specific frequency and time. Narrow-band signals that are on for the duration of a short observation appear as lines across waterfall plots, which may be sloped due to the relative motion between the celestial source and the telescope, the so-called Doppler acceleration (Sheikh et al. 2019). If a signal is bright enough, it is easily distinguishable by the human eye. However, it is simply impossible to visually inspect all the data we collect, which easily spans billions of frequency channels (Lebofsky et al. 2019).

Our standard narrow-band signal search method uses TurboSETI,[9] an implementation of the "tree deDoppler" algorithm, which effectively averages along potential Doppler drift rates (slopes) in a spectrogram and searches for statistically high spikes in the resulting spectra (Taylor 1974; Siemion et al. 2013; Enriquez et al. 2017; Enriquez & Price 2019). If one picks the correct drift rate and there is a signal at that rate, one should get a detection, since averaging reduces the impact of random noise and preserves the signal. While the underlying tree-based algorithm is more efficient than a naive search over all drift rates, this approach requires many passes over the data and potentially misses fainter signals masked by bright RFI (Pinchuk et al. 2019).

A complementary parametric algorithm for localizing narrow-band signals is the Hough transform, an edge-detection technique that translates an image into another 2D representation whose features correspond to edges in the original image (Hough 1959; Barinova et al. 2012). Applying this transform to Stokes-*I* data and identifying bright features allows for the detection and localization of narrow-band signals, which manifest as "edges" in the data (Monari et al. 2006; Fridman 2011). This method also requires many passes over the data, however, and the features must still be extracted from the resulting transform (e.g., via thresholding).

Another method for detecting signals in radio data is to analyze the degree to which the data differs from an ideal statistical distribution, assuming only noise is present. For instance, higher order statistics such as the kurtosis can indicate that a portion of data differs significantly from an ideal Gaussian distribution. Applying this principle by calculating the kurtosis for time series voltage data or the spectral kurtosis for dynamic spectra can signal the presence of RFI (Ruf et al. 2006; Nita et al. 2016). Since these are relatively simple calculations, they can be done in real-time to flag or even mitigate RFI during observations.

While these approaches each have their own strengths, we would like to evaluate the effectiveness of machine learning (ML) methods in accurately identifying narrow-band signals, especially in the presence of bright RFI. Having a good signal localization and detection pipeline is crucial for identifying signals that are currently overlooked using conventional signal processing methods.

Advances in computer vision techniques, especially with convolutional neural networks (CNN), have proven quite effective in classification and object detection tasks (Krizhevsky et al. 2012; Simonyan & Zisserman 2014; Ren et al. 2015; Szegedy et al. 2015; He et al. 2016). For radio signal processing, CNNs have been used to classify radio transmissions based on their modulation schemes, using time series voltage data (O'Shea et al. 2018). In fact, Stokes-*I* data produced from radio observations is similar in structure to an image, so the data lends itself readily to computer vision techniques.

For instance, Zhang et al. (2018) used CNNs on Stokes-*I* data to detect new pulses from the fast radio burst FRB 121102. They created a data set of synthetic FRB pulses and trained a classifier to detect whether or not a data frame contained a pulse. Using this model on real observations, they were able to find 72 new pulses within a 5 hr radio observation.

Similarly, Harp et al. (2019) created a synthetic data set of radio spectrograms to resemble data taken by the Allen Telescope Array, inserting 6 different classes of narrow-band signals into an artificial noise background. They compared the performance of various deep CNN architectures in classifying the synthetic data frames, and found that their ML classifiers performed well for signals with relatively high signal-to-noise ratios (S/N).

Modern object detection methods such as You Only Look Once (YOLO; Redmon et al. 2016) use clever ways to quickly

[9] https://github.com/UCBerkeleySETI/turbo_seti

determine an arbitrary number of object bounding boxes in images. Even so, object detection and localization of long, thin objects remain particularly difficult. It is hard to draw meaningful bounding boxes around them, since such objects generally comprise only a small portion of bounding box areas, making it impractical to maximize the intersection over union measure with ground truth. In addition, since many radio signals can intersect at any position, it is harder to similarly split up an image frame into a coarse grid and only associate one signal with each grid cell, as in YOLO. This makes it especially difficult to detect an arbitrary number of signals in a frame. For this reason, we limit our present work to signal localization, in which we attempt to precisely predict the positions of a known number of signals in each image frame.

In this work, we investigate the effectiveness of ML signal localization on synthetic radio spectrogram data. We run experiments using CNN architectures and evaluate performance based on the root mean square error between true and predicted pixel locations as a function of signal intensity or S/N. We further compare these localization results with signal detections from TurboSETI. We conclude with future directions for improving signal localization and ultimately moving toward true object detection.

2. Data and Preprocessing

The SETI goal of looking for interesting signals in observations makes it difficult to get a large labeled data set. To that end, it is an open question as to what sort of labels make the most sense—there are so many different forms and patterns in human RFI that results would be highly dependent on the number and nature of classes. Furthermore, manual inspection can be ineffective in identifying lower intensity signals (whereas averaging along various drift rates can increase the S/N and thus reveal dimmer signals).

To test the sensitivity and accuracy of signal search procedures, we generated a set of synthetic observations that resemble real data from the GBT. In general, the BL instrument at the GBT takes data over a range of frequencies (over a large bandwidth of a few GHz) at the same time (MacMahon et al. 2018). Here, we focus on scientific data products that have a 1.4 Hz spectral resolution and a 1.4 s temporal resolution, at a frequency range of 4–8 GHz (*C*-band).

For this work, we analyzed image frames that are 32 × 1024 pixels—32 time samples tall and 1024 frequency samples wide. This effectively spans a total range of about $32 \times 1.4\ \mathrm{s} \approx 45$ s and $1024 \times 1.4\ \mathrm{Hz} \approx 1430$ Hz. Although our observations easily span billions of frequency channels, for practical reasons, we limit the number of frequency channels per frame to better facilitate the use of CNNs.

Table 1
Parameters for Sgr B2 Data

Parameter	Sgr B2
R.A. (J2000)	$17^{\mathrm{h}}47^{\mathrm{m}}15\overset{\mathrm{s}}{.}0$
Decl. (J2000)	$-28°22'59\overset{''}{.}16$
Initial MJD	58465.71709
C-Band Frequency Coverage	6564–6752 MHz
Frequency Resolution	1.39698 Hz
Time Resolution	1.43166 s
Integration Time	60 s

2.1. Noise Properties

Since the background radiometer noise in time-series voltage data closely follows a zero-mean Gaussian distribution, the noise in Stokes-*I* data ultimately follows a chi-squared distribution (McDonough & Whalen 1995; Nita et al. 2007; Thompson et al. 2017). However, due to instrumental effects such as coarse channel bandpass shapes and natural variations in detector sensitivity as a function of frequency, the raw intensity values we get from observations can vary appreciably.

To properly capture these intrinsic intensity variations in our synthetic data, we based their noise properties on actual data from the GBT. We used 4–8 GHz observations of Sgr B2 taken on 2018 December 13 at 17:12:37 UTC. We found that using a smaller range of frequencies, 6564–6752 MHz, was sufficient for obtaining realistic background intensity values for the synthetic frames (Table 1).

At the GBT, the Versatile Green Bank Spectrometer (Prestage et al. 2015) digitizes and coarsely channelizes data using a polyphase filterbank. The data is then sent to the BL data recorder system (MacMahon et al. 2018), which applies finer channelization to each coarse channel and records the resulting high spectral resolution data products.

Figure 1 shows the integrated bandpass plot of our observational data. Visible in the spectrum are the 64 coarse channels present in our data slice, which are about 3 MHz in width and characterized by intensity fall-offs on either edge. The spike at the center of each coarse channel is the so-called "DC bin," the sum of all samples within the channel, which arises from the Fourier Transform-based filterbank. The large spike at ∼6670 MHz is bright RFI. The overall bandpass shape reflects the inherent variation in sensitivity across the receiver.

We split this data into frames of size 32 × 1024 and calculate the mean intensity of each individual frame. Since some frames contain DC bins and bright RFI that bias these intensities toward higher values, we trim the resulting collection of mean intensities using sigma clipping with limits corresponding to 5σ. Figure 2 shows a histogram of these intensities after cutting out outliers, for a total of 126,419 remaining frames. Together with Figure 1, we notice that the

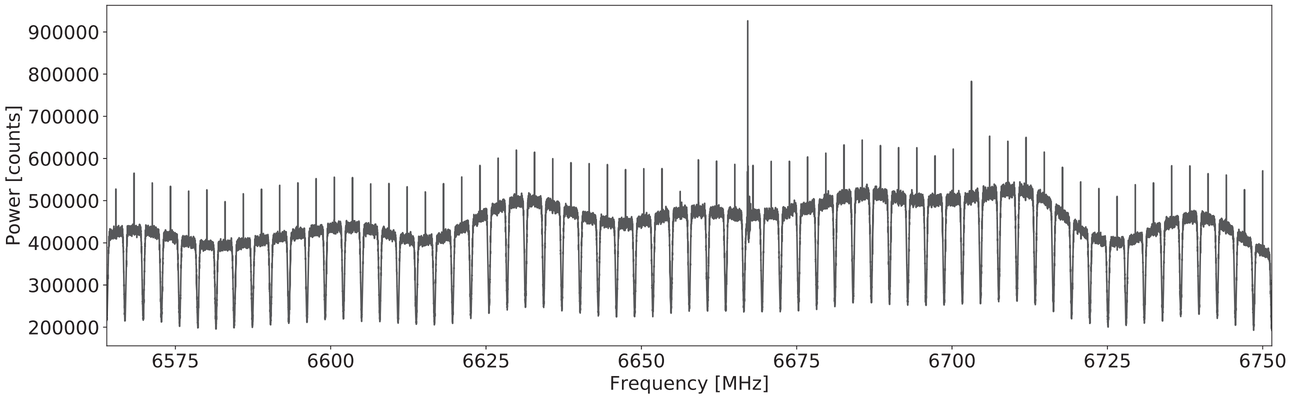

Figure 1. Bandpass plot for Sgr B2 data over an integration time of 60 s.

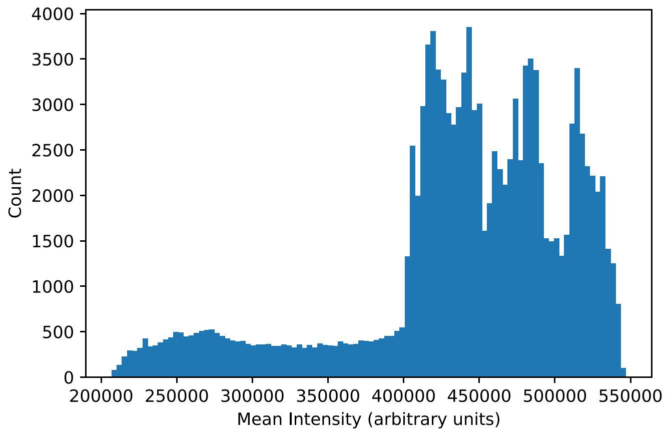

Figure 2. Histogram of mean frame intensities over real GBT observation after trimming outliers, for a total of 126,419 samples.
(A color version of this figure is available in the online journal.)

majority of mean intensities are concentrated between 4×10^5 and 5.5×10^5. The tail extending down to intensities of 2×10^5 is due to the lower sensitivities at the edges of each coarse channel. Note that each frame only covers ∼1.4 kHz in frequency, which is small in comparison to coarse channels (∼3 MHz). So, we make the assumption that larger scale systematic bandpass effects are not important within individual frames.

To initialize each synthetic data frame with noise, we randomly select from this empirical distribution to select a desired mean intensity μ_{noise}. To calculate the degrees of freedom for our chi-squared noise background, we note that our Stokes-*I* data uses two polarizations and that complex Fourier coefficients contribute both real and imaginary terms. These result in 4 degrees of freedom for every integration, so overall, the underlying distribution has a total of $k = 4 \cdot df \cdot dt$ degrees of freedom, where df and dt are the frequency and time resolutions of the data. Since the mean of a chi-squared distribution is k, we randomly sample values from this distribution to populate the empty data frame, and then scale every value up by a factor of μ_{noise}/k to match the desired mean intensity. Comparing the results to our observations, we find that this procedure indeed reproduces the noise distributions found in real data. The benefit in having a method for generating purely synthetic yet realistic background noise is that every data frame thus created is guaranteed to be free from signals of any kind, which helps in accurately evaluating signal search strategies.

2.2. *Synthetic Signals*

Narrow-band signals found in our radio frequency data come in many forms, with temporal and spectral structures of varying complexity. For example, modulation schemes in prolonged radio transmissions result in intensity variations over time, and some signals are emitted in short pulses in the first place (Sokolowski et al. 2015). Depending on their location in the galaxy and in the sky, narrow-band signals may also be subject to scintillation from the interstellar and interplanetary medium (Rickett 1977; Lotova et al. 1985; Cordes et al. 1997; Siemion et al. 2013; Price et al. 2019). Even a constant amplitude sine wave signal in the time domain, in the presence of zero-mean Gaussian noise, follows a non-central chi-squared distribution of intensities in Stokes-*I* data (McDonough & Whalen 1995). Examples of various RFI morphologies are presented in Sheikh et al. (2020).

Despite the prevalence of complex and noisy narrow-band signals, for this work, we choose to create synthetic signals with constant intensity over time as heuristic models for real signals. Indeed, this makes the assumption that if an ML model can accurately localize these ideal signals, it will also properly localize more complex signals. Our intuition behind this assumption stems from the fact that search techniques such as TurboSETI will still find noisy signals, even though they are optimized for finding "simple" signals.

We developed a software package, Setigen,[10] to facilitate the creation and injection of synthetic narrow-band signals into

[10] https://github.com/bbrzycki/setigen

observational data frames. Based on the time and frequency resolutions of the Stokes-*I* data, Setigen can also calculate the corresponding idealized chi-squared distribution for the background noise, into which synthetic signals can be added, as described in Section 2.1. This allows us to create large data sets of synthetic data frames for training and validating signal search pipelines.

We define the "start" of a signal as the index (or pixel) in the frequency direction where the center of the signal is during $t = 0$ in an image frame, and the "end" as the center position during the 32nd time sample. We randomly choose the starting and ending indices for each signal and the width of the signal in the frequency direction (limited to a narrow-band range). Starting and ending indices are always between 0 and 1023, inclusive. Accordingly, the maximum absolute drift rate, corresponding to starting and ending indices on opposite ends of the frequency range, would be about 31 Hz s^{-1}.

Because we would like to analyze the effectiveness of our ML algorithms on different S/N levels, we scale the intensity of each synthetic signal according to the desired S/N level, the mean background noise level, and the number of time samples:

$$\mathrm{S/N} = \frac{I_{\mathrm{signal}}}{\mu_{\mathrm{noise}}} \times \sqrt{n_t}, \tag{1}$$

where I_{signal} is the appropriate intensity of the injected constant signal at any single time samples, μ_{noise} is the mean of the background noise, and n_t is the number of time samples (in this case, $n_t = 32$).

This definition is used so that the expected S/N matches the measured S/N if we had simply averaged through each time sample shifted at the correct drift rate, which is how current Doppler drift search pipelines, such as TurboSETI, work (Enriquez et al. 2017).

2.3. Data Set Construction

We generate two data sets to test signal localization, each with 120,000 training samples and 24,000 test samples. Since the signals we are interested in potentially span a large range of intensities, we specify S/N levels for our synthetic narrow-band signals using decibels, such that 0 dB $\rightarrow 1\sigma$, 20 dB $\rightarrow 100\sigma$, etc.

Our first data set contains 32 × 1024 image frames with exactly one signal at S/N levels of 0, 5, 10, 15, 20, and 25 dB. So, for each S/N level, we generate 20,000 training frames and 4000 test frames. For each frame, we also save the starting and ending indices (2 numbers) as labels.

Our second data set contains frames with exactly two signals. One of the signals is 25 dB and at a zero drift rate, so that it is at a constant frequency at all time samples. This is meant to represent a typical RFI-like signal. The other signal is at S/N levels of 0–25 dB as in the first data set. We save the starting and ending indices for both signals (4 numbers). Note that these labels *are* ordered, even though there is no preferred order in any given image frame.

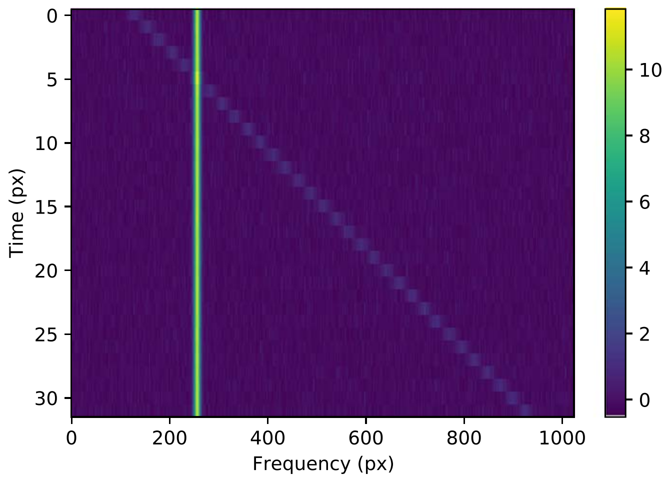

Figure 3. Synthetic data frame with two signals, one "RFI" signal at 25 dB and zero drift, and one dimmer signal at 15 dB, normalized over the entire frame to mean 0 and variance 1.
(A color version of this figure is available in the online journal.)

2.4. Preprocessing

By design, for each data set, we generate 120,000 training frames and 24,000 test frames in total, the latter of which are used only for final evaluation. During training, we take a 80/20 random split of the training frames for training/validation.

The choices of normalization are very important for our data, which can exhibit regions of high contrast and varying instrument sensitivity. We choose to normalize our labels (indices between 0 and 1023 inclusive) to be between 0 and 1 by dividing out by 1024. Normalizing our input data frames is more interesting, and there are multiple potential ways to go about this.

The first would be to normalize over an entire frame by subtracting the mean and dividing by the variance over all pixels, so that our normalized frame has mean 0 and variance 1, as in Figure 3. Another method useful in astronomy is normalizing by frequency, where we subtract the mean and divide by the variance in the time direction for each frequency sample. This also yields mean 0 and variance 1, but serves to specifically normalize out differences in instrument sensitivity as a function of frequency. Normalizing over an entire frame preserves these sensitivity differences.

However, lots of detected narrow-band signals are RFI and thus moving with the Earth, so they do not exhibit Doppler accelerations and appear as vertical lines in waterfall plots. So if we normalize by frequency, a constant vertical signal will disappear from the data, since we subtract out the average (constant) intensity, as in Figure 4. Since we are interested in localizing all signals and eventually comparing our ML methodology with standard search techniques, we certainly

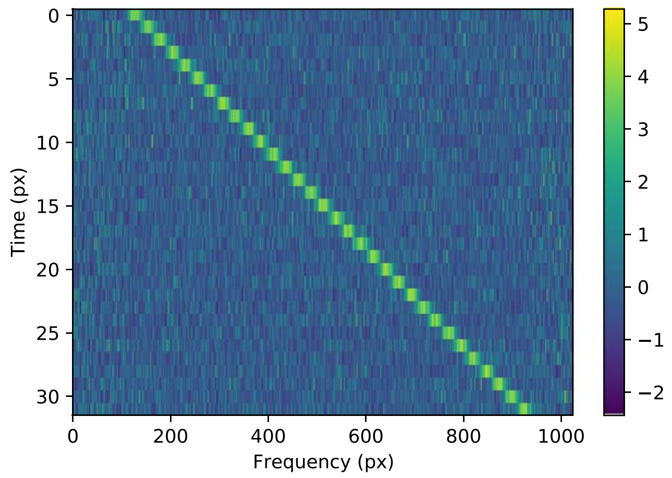

Figure 4. Data frame containing the same data as Figure 3, instead normalized per frequency channel to mean 0 and variance 1.
(A color version of this figure is available in the online journal.)

do not want to exclude this information via our normalization procedure. On the other hand, this can potentially strengthen our sensitivity toward dimmer, sloped signals.

Considering these idiosyncrasies in our data, we test both of these two normalization methods as inputs into our models.

3. Methods

We present the signal search methods used in this work. Namely, we discuss the CNN architectures explored for ML-based localization and briefly describe our standard deDoppler algorithm, TurboSETI.

3.1. CNN Model Architectures

In this work, we define both a "baseline" and a "final" model. We take our baseline architecture to be a simple CNN that is typically used on general image classification tasks. Our final model contains architectural improvements influenced by the nature of our data and training tasks. Specifically, we compare the baseline and final models and evaluate the extent to which the architectural changes improve localization accuracy.

For our two data sets, we use the same overall model architecture except for the final regression layer, which either has 2 or 4 nodes depending on whether we are predicting the position(s) of one or two signals. Since we would like to predict each signal position as best as possible, where the position is defined by its starting and ending indices, we seek to minimize the mean squared error between true and predicted indices.

Our models are implemented using the Keras functional API (Chollet et al. 2015). The source code for generating our data sets and training these models is available at github.com/bbrzycki/seti-nb-localization.

3.1.1. Baseline Model

For our baseline model, we choose a simple CNN with 4 convolutional layers, 3 max pooling layers, 2 fully connected layers (each with 64 nodes), and a dropout layer at 50%. Our input is a single 32×1024 data frame, normalized over the entire frame. This is a typical architecture for image classification tasks, making it a good baseline for comparison. We use rectified linear unit (ReLU) activations after each convolutional and fully connected layer. For the two signal task, this model has 1,073,988 trainable parameters. Figure 5 shows the baseline model architecture in detail.

3.1.2. Final Model

For our final model, we have 2 residual connections, 5 convolutional layers in total (using stride 2 instead of max pooling), 2 fully connected layers (both with 1024 nodes), and a dropout layer at 50%. We again use ReLU activations after each convolutional and fully connected layer, as well as batch normalization after summing frames in residual connections. Although our image frames were normalized to a mean of 0 and therefore contained negative numbers, we found that alternate activation functions to ReLU, such as tanh, did not improve localization accuracy.

Residual connections are marked by shortcuts between convolutional layers; in our case, we use element-wise addition between a given layer and a following convolutional layer (He et al. 2016). These connections reduce over-fitting and enhance accuracy by counteracting vanishing gradients in neural networks. Furthermore, since narrow-band signals are thin relative to our image frames, residual connections allow thinner features to propagate further into our models. We follow up these additive connections with batch normalization layers to ensure that the lower-order statistics of layer inputs at these positions in our model remain the same across batches of data (Ioffe & Szegedy 2015).

For our inputs, we express the data frames as "images" with two channels—one channel is the data frame normalized over the entire frame, and the other is the data frame normalized per frequency. Our rationale for using a two channel input is that when one normalizes over the entire frame, the model finds the brighter signals much more easily, and the dimmer signals could be washed out. However, most of time in radio data, the brightest signals are also at zero drift rate, since they originate from Earth. Therefore, normalizing by frequency could serve to remove these brightest signals and show more sensitivity to dimmer, drifted signals. Using both forms of image normalization as inputs into the same model could help better identify these different forms of signals appearing in our data.

For the two signal task, this model has 26,070,916 trainable parameters. Figure 6 shows the final model architecture in detail.

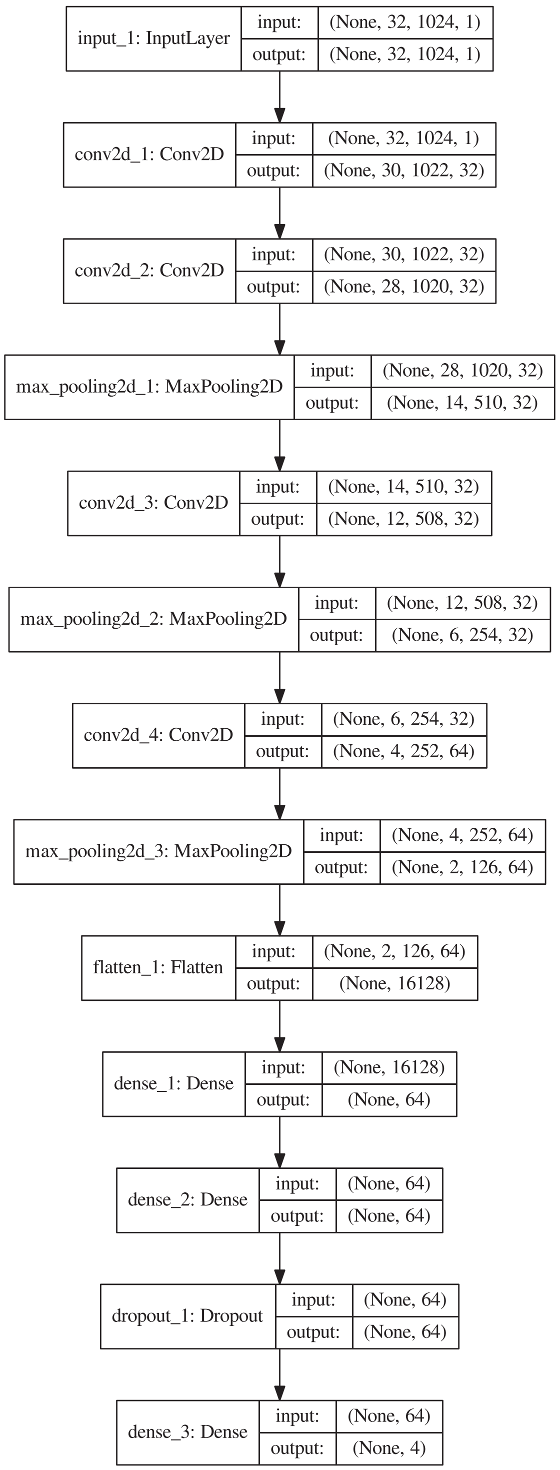
input_1: InputLayer | input: | (None, 32, 1024, 1) | output: | (None, 32, 1024, 1)
conv2d_1: Conv2D | input: | (None, 32, 1024, 1) | output: | (None, 30, 1022, 32)
conv2d_2: Conv2D | input: | (None, 30, 1022, 32) | output: | (None, 28, 1020, 32)
max_pooling2d_1: MaxPooling2D | input: | (None, 28, 1020, 32) | output: | (None, 14, 510, 32)
conv2d_3: Conv2D | input: | (None, 14, 510, 32) | output: | (None, 12, 508, 32)
max_pooling2d_2: MaxPooling2D | input: | (None, 12, 508, 32) | output: | (None, 6, 254, 32)
conv2d_4: Conv2D | input: | (None, 6, 254, 32) | output: | (None, 4, 252, 64)
max_pooling2d_3: MaxPooling2D | input: | (None, 4, 252, 64) | output: | (None, 2, 126, 64)
flatten_1: Flatten | input: | (None, 2, 126, 64) | output: | (None, 16128)
dense_1: Dense | input: | (None, 16128) | output: | (None, 64)
dense_2: Dense | input: | (None, 64) | output: | (None, 64)
dropout_1: Dropout | input: | (None, 64) | output: | (None, 64)
dense_3: Dense | input: | (None, 64) | output: | (None, 4)

Figure 5. Baseline model architecture for the two signal localization task. For each block, the layer type is shown, along with input and output shapes. Inputs have shape 32 × 1024 × 1, normalized over all pixels. These are passed through an initial convolutional layer, followed by 3 pairs of convolutional and max pooling layers. This is followed by two fully connected layers and a dropout layer, before finally going into the output layer. For the single signal task, the last layer has 2 nodes instead of 4.

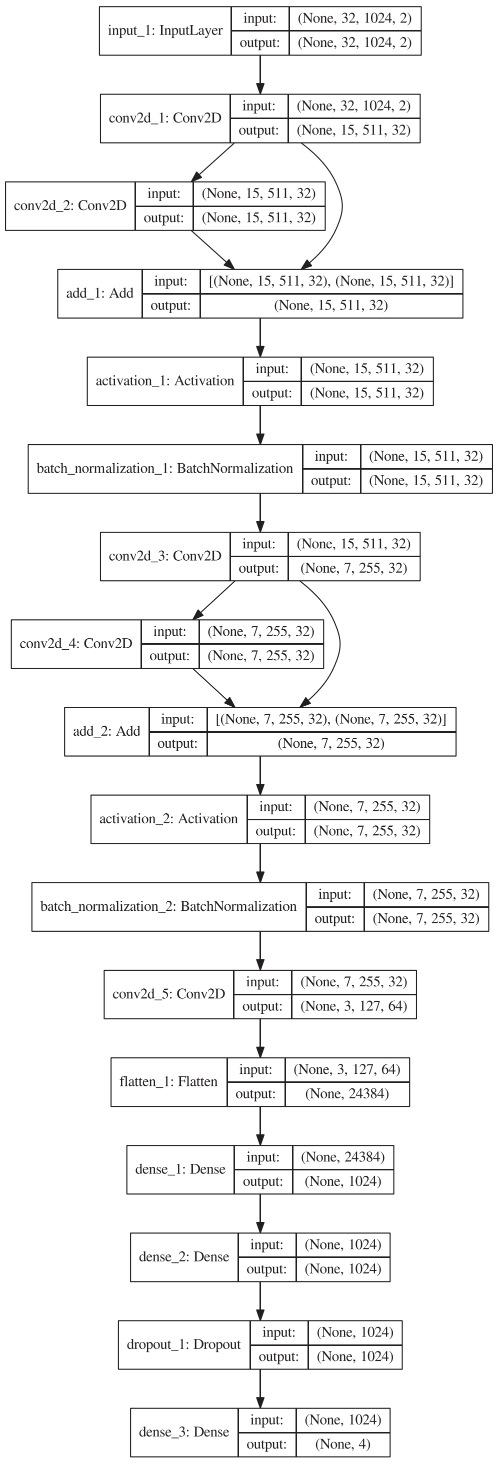
input_1: InputLayer | input: | (None, 32, 1024, 2) | output: | (None, 32, 1024, 2)
conv2d_1: Conv2D | input: | (None, 32, 1024, 2) | output: | (None, 15, 511, 32)
conv2d_2: Conv2D | input: | (None, 15, 511, 32) | output: | (None, 15, 511, 32)
add_1: Add | input: | [(None, 15, 511, 32), (None, 15, 511, 32)] | output: | (None, 15, 511, 32)
activation_1: Activation | input: | (None, 15, 511, 32) | output: | (None, 15, 511, 32)
batch_normalization_1: BatchNormalization | input: | (None, 15, 511, 32) | output: | (None, 15, 511, 32)
conv2d_3: Conv2D | input: | (None, 15, 511, 32) | output: | (None, 7, 255, 32)
conv2d_4: Conv2D | input: | (None, 7, 255, 32) | output: | (None, 7, 255, 32)
add_2: Add | input: | [(None, 7, 255, 32), (None, 7, 255, 32)] | output: | (None, 7, 255, 32)
activation_2: Activation | input: | (None, 7, 255, 32) | output: | (None, 7, 255, 32)
batch_normalization_2: BatchNormalization | input: | (None, 7, 255, 32) | output: | (None, 7, 255, 32)
conv2d_5: Conv2D | input: | (None, 7, 255, 32) | output: | (None, 3, 127, 64)
flatten_1: Flatten | input: | (None, 3, 127, 64) | output: | (None, 24384)
dense_1: Dense | input: | (None, 24384) | output: | (None, 1024)
dense_2: Dense | input: | (None, 1024) | output: | (None, 1024)
dropout_1: Dropout | input: | (None, 1024) | output: | (None, 1024)
dense_3: Dense | input: | (None, 1024) | output: | (None, 4)

Figure 6. Final model architecture for the two signal localization task. Inputs have shape 32 × 1024 × 2, combining the two normalizations described in Section 3.1.2. Residual connections are apparent between convolutional layers, followed by a batch normalization layer. This structure is repeated twice, followed by another convolution layer, two fully connected layers, and a dropout layer. For the single signal task, the output layer has 2 nodes instead of 4.

3.2. TurboSETI

To provide a standard for performance comparison, we also ran our one signal data set through the TurboSETI suite. As mentioned before, TurboSETI uses a tree-based deDoppler algorithm to efficiently search for signals above a specified S/N threshold over a specified range of drift rates (Enriquez et al. 2017; Enriquez & Price 2019). For each detected signal, TurboSETI returns the starting index/frequency and drift rate, along with the calculated S/N.

Previous studies have used the detection threshold of an S/N of 10; going any lower results in an unacceptable number of false positive detections (Price et al. 2019; Sheikh et al. 2020). We likewise set a detection threshold at an S/N of 10, or 10 dB.

Selecting a maximum Doppler drift rate range to search presents a trade-off between potential detections and computational time. Sheikh et al. (2020) uses a maximum absolute drift rate of 20 Hz s^{-1}, which is the largest thus far for a deDoppler search strategy. As mentioned in Section 2.2, the largest possible absolute drift rate in our 32×1024 frames is 31 Hz s^{-1}, so we simply choose that as our maximum search drift rate.

Because TurboSETI works by integrating along straight line paths, it struggles to find dim signals that are close to or intersect brighter signals. We observe this with every frame in our two signal data set, in which TurboSETI only finds the bright, non-drifted "RFI" signal. Accordingly, we only compare our TurboSETI search results with the ML predictions over the one signal data set.

4. Results

4.1. Baseline versus Final Model

For the baseline and final model architectures, we compare the error between true and predicted indices as a function of S/N. To get a more intuitive feel on model performance, we calculate 1024× RMSE, where RMSE is the root mean square error, to see the errors in units of pixels/indices:

$$\text{RMSE (index units)} = 1024 \times \sqrt{\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2}, \qquad (2)$$

where n is the number of indices in our labeled data (2 and 4 for one and two signal data sets), y_i are predicted indices, and $\hat{y}_i$ are true indices.

We first train our models on the full 0–25 dB S/N levels at each data set. We also analyze the effect of only training on frames with a 10–25 dB signal, cutting out the 0 and 5 dB levels. Here, we again use a train-test split of 80/20 out of this restricted training set. However, we still evaluate these trained models on the full test set. Results from the restricted 10–25 dB data set are labeled "bright" in Figures 7–8.

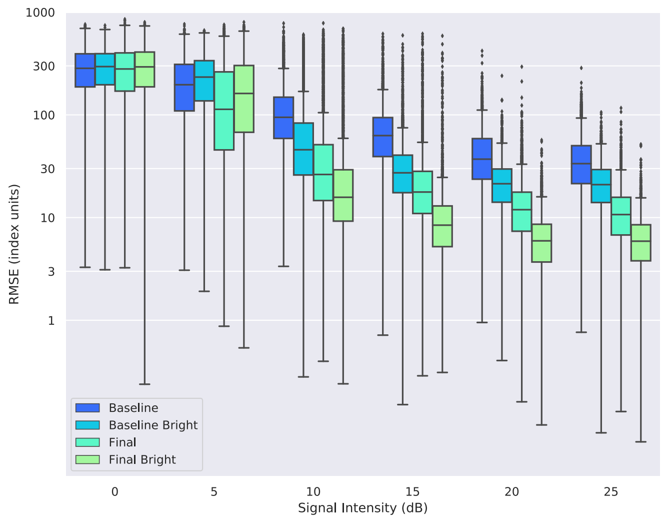

Figure 7. Box plot of RMSE in index/pixel units for the one signal data set as a function of signal S/N. We compare the final and baseline models trained on both the full 0–25 dB data set, as well as the truncated "bright" 10–25 dB data set.
(A color version of this figure is available in the online journal.)

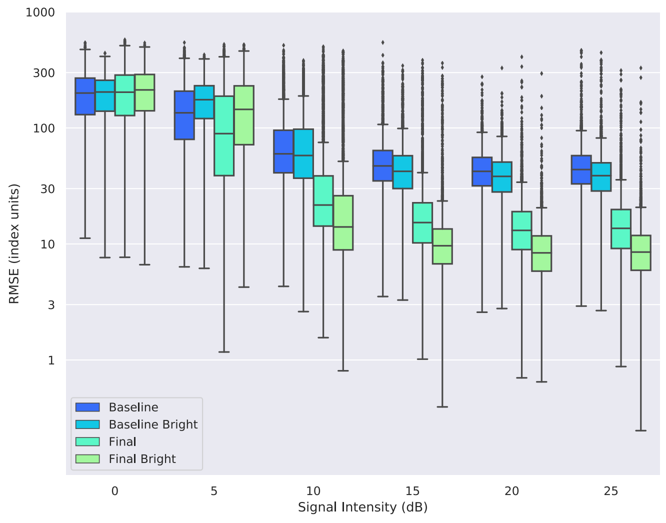

Figure 8. Box plot of RMSE in index/pixel units for the two signal data set as a function of signal S/N. We compare the final and baseline models trained on both the full 0–25 dB data set, as well as the truncated "bright" 10–25 dB data set.
(A color version of this figure is available in the online journal.)

In these figures, we plot a box and whisker plot (with outliers above and below the median by 1.5 times the spread between 25% and 75% quartiles) for each S/N level and each training run for our baseline and final models on the full and bright data sets. For each case, we have outliers with high errors that would tend to bias our evaluations much higher if we only consider the mean RMSE.

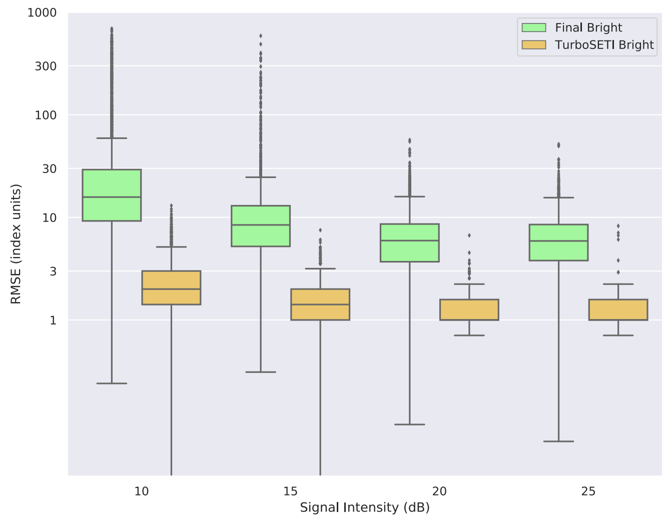

Figure 9. Box plot of RMSE in index/pixel units, comparing performance of our final model architecture with TurboSETI on the one signal data set as a function of signal S/N.
(A color version of this figure is available in the online journal.)

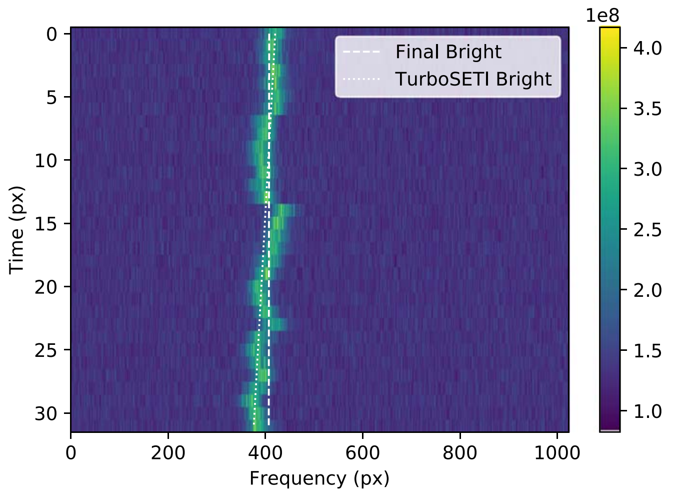

Figure 10. Observational data frame containing an RFI signal, overlaid with localization predictions from our final CNN architecture (dashed) and TurboSETI (dotted). Although the real signal is more complex than those in our training data, the model produces reasonable predictions.
(A color version of this figure is available in the online journal.)

4.2. TurboSETI versus Final Model

For each frame in the one signal data set, we use TurboSETI to get signal localization results, which we translate into starting and ending indices. This allows us to calculate the RMSE in index units, exactly as we did for the ML predictions. Figure 9 shows the results, again as a function of S/N level, compared to predictions from our final CNN model trained on "bright" frames. Note that we only compare predictions down to an intensity of 10 dB, since that is our TurboSETI detection threshold.

We also compare the computational costs of each search method. For the final CNN model, generating predictions for all 24,000 frames in the test set takes about 70 s using an Nvidia Titan Xp GPU with a batch size of 1. Using a batch size of 32, generating all predictions takes about 34 s.

Although we run TurboSETI with a 10 dB threshold, for bench-marking purposes, we ran it on all 24,000 frames in the test set. This takes a total of 2.6 hr using a single CPU. In practice, however, TurboSETI is run on large data frames, on order of billions of frequency channels, as opposed to the comparatively small frames used in this work. This is because the tree-based search algorithm is most efficient when applied to a few large files, as opposed to many smaller ones. To make a fairer comparison, we ran TurboSETI on a large frame with the same amount of data as if all 24,000 test frames were concatenated along the frequency axis. Using the same search parameters of a 10 dB threshold and a maximum 31 Hz s^{-1} drift rate, TurboSETI takes about 20 minutes to finish searching this concatenated data frame. Making predictions using the CNN model is therefore more efficient, especially when using larger batch sizes.

Lastly, we tested both TurboSETI and the final CNN model on localizing a few known RFI signals in the *C*-band observation described in Section 2.1. Figure 10 shows an example of predicted localization paths from both methods for a complex RFI signal. As expected, TurboSETI generates the best-fit localization, since it integrates along each potential drift rate. Nevertheless, the final model gives a reasonable prediction, despite being trained on idealized signals with constant intensity and constant drift rate.

5. Discussion

For the full training data sets, the baseline and final models show a smooth progression of better median RMSE values from 0 to 25 dB. Our final model seems to outperform the baseline model consistently, particular for signals of at least 10 dB, on the order of a 3× reduction in error for both one and two signal data sets (Figures 7–8).

Unsurprisingly, for both one and two signal data sets, the signals at 0 dB (S/N = 1; 1σ) do very poorly, with an typical error of 200–400 pixels off. As a check, note that the average expected distance between two randomly chosen points on a line segment of length L is $L/3$. For 1024 total pixels, this means on average, selecting an index at random as a prediction should yield errors of about $1024/3 \approx 341$ pixels. So for 0 dB signals, our predictions are essentially random. In a way, this is perfectly acceptable, since in general we do not accept 1σ as a true detection of a signal in the first place.

When we compare these results with those from models only trained on 10–25 dB frames, we see a few interesting things. As expected, the "bright" models then perform worse on the lower S/N signals, 0 and 5 dB. For 0 dB signals, we still get effectively random predictions, but the decrease in performance at 5 dB is notable. However, this restricted training set improves the performance at 10–25 dB appreciably, in different ways. In the one signal case, the final model outperforms the baseline model by about 3× for all S/N levels of at least 10 dB. In the two signal case, frames with at least 15 dB signals improved about 5× between final and baseline models. We further observe that, in each case, restricting training to frames with at least a 10 dB signal improves overall model performance compared to training with the full data set. Since both models seem to only be capable of nearly random predictions at the lowest S/Ns, doing training steps on that data tends to hurt the performance at higher S/Ns.

The best performing models were the final model trained on the bright data set. At an S/N level of 25 dB, the one and two signal cases reached a median of about 6 and 9 pixels of RMSE, respectively. We expect the one signal models to do better than the two signal models, since there is only a single bright signal to try to localize. Nevertheless, it was surprising that even our best models did not consistently localize either of these cases to extremely high precision, i.e., median RMSE of about 1 pixel. On the one hand, localizing a bright signal to a precision of 6 pixels out of 1024 is decent (corresponding to about 8.4 Hz), but on the other, we expected to do even better, since we have synthetic data sets and therefore know *precisely* where the center of each signal lies. Instead, as we observe in Figure 9, TurboSETI very accurately localizes these signals, to a median RMSE of about 1–2 pixels for each S/N.

For the ML predictions, there are still large outliers, even reaching levels of randomness (300–500 pixels off) in the two signal case. One potential explanation and limitation in our model is that our labels are ordered. For two signals, at the highest S/N, both signals are at 25 dB. The model could have a harder time differentiating one signal for another (where the only distinguishing factor is zero versus non-zero drift rate) and produce bad predictions. Our test set has 4000 images for each S/N level, so even having a small fraction of these show up as outliers at the highest S/Ns would be compounded if we tried using this model on real data. Recall that a 32×1024 px image at 1.4 s and 1.4 Hz resolution is about 45 s by 1430 Hz in total. For a typical 5 minute *C*-band observation (4–8 GHz), this is equivalent to about 18.65 million data frames. Before it makes sense to use our CNN pipeline in searches on real data, we need to find ways of very precisely localizing these signals so that we are not swamped with false positives and inaccurate positions.

Nevertheless, inspecting ML predictions in the two signal case revealed that the models appear to learn that the first two labels (i.e., corresponding to the starting and ending index of the zero drift RFI signal) should be the same index, since it predicted essentially the same value between [0, 1) up to a few significant figures (at least to differences of $1/1024$).

Despite being much more accurate over the one signal data set, TurboSETI takes much longer to produce localization predictions than our CNN models for the same amount of data, on the order of 20 minutes versus 34 s, as discussed in Section 4.2. Of course, this is not necessarily surprising, since our ML predictions take advantage of GPU-accelerated calculations, especially when we batch together multiple data frame inputs at once.

For every frame in the two signal data set, we also observe that TurboSETI struggles to find multiple signals that are close together or intersecting, and instead only detects the brighter one. For the same data set, our final CNN model can generally localize such signals to an RMSE of about 10–20 pixels for signals at 10 dB and above. Besides being more computationally efficient, CNN-based pipelines may therefore be better at finding elusive signals that standard search techniques tend to miss.

In addition, it is encouraging that when we take into account the multiple possible normalizations, our model performance improves, especially in the two signal case with a model RFI signal. We believe that this could generalize well to frames with over two signals, as long as we can find a good way of matching labels and predictions, perhaps without necessarily enforcing an ordering.

6. Summary

Accurately identifying the presence and positions of signals in radio data is important for finding candidate technosignatures and ensuring that we do not miss interesting signals in the presence of bright RFI. Computer vision techniques allow us to ingest complex image frame data and distill them into relevant information, such as signal locations.

We found that our final model outperformed our baseline model for all S/Ns in both data sets, and training each model on data sets limited to 10–25 dB results in significant increases in model performance. Our best results had a median RMSE of about 6 pixels in the one signal case and about 9 pixels in the two signal case. Since we used simple signals embedded in ideal chi-squared noise, we expected our localization models to perform even better, especially in the one signal case. Nevertheless, while these errors are higher than expected and come with a host of outliers that perform much more poorly, these results are promising for future work in localizing narrow-band signals in images.

We also did an analysis using TurboSETI to detect signals in our synthetic data sets, and compared the results with our ML predictions. We found that while TurboSETI produces more accurate localizations for the one signal data set, our ML pipelines are much faster and able to produce meaningful results even for more complex tasks, such as localizing both signals in our two signal data set.

Overall, object detection and localization of long, thin objects is difficult since they do not match the typically rectangular shape of many other objects, and so it is harder to maximize intersection over union measures with ground truth. Although on the one hand detecting lines seems simple intuitively, the relative lack of information compared to broader, extended objects makes it more difficult, especially within a noisy background. Nevertheless, with a few key assumptions that are specific to the radio data we collect for SETI, we can make more progress in precisely localizing radio frequency signals.

A future direction for this work is to investigate the effectiveness of treating multiple normalizations as independent inputs to the same model, instead of combining them as a single two channel input. Each normalization could have a few convolutional layers to itself, and would be added to each other to learn features with contributions from both normalizations. Indeed, this approach could scale better and benefit from additional data preprocessing techniques beyond the two normalizations discussed in this paper.

We can also easily use this CNN architecture to classify signals, or to both classify and localize simultaneously depending on how we choose our labels and loss functions. We are interested to see how well this method extends to more than two signals in a single image frame, and eventually, we would like to develop a pipeline for signal *detection* of an arbitrary number of signals in a given image frame.

Breakthrough Listen is managed by the Breakthrough Initiatives, sponsored by the Breakthrough Prize Foundation. The Green Bank Observatory is a facility of the National Science Foundation, operated under cooperative agreement by Associated Universities, Inc. We thank the staff at the Green Bank Observatory for their operational support.

Software: Keras (Chollet et al. 2015), Setigen,[11] TurboSETI (Enriquez & Price 2019), Blimpy (Price et al. 2019), Matplotlib (Hunter 2007).

ORCID iDs

Bryan Brzycki https://orcid.org/0000-0002-7461-107X
Andrew P. V. Siemion https://orcid.org/0000-0003-2828-7720
Steve Croft https://orcid.org/0000-0003-4823-129X
Daniel Czech https://orcid.org/0000-0002-8071-6011
David DeBoer https://orcid.org/0000-0003-3197-2294
Vishal Gajjar https://orcid.org/0000-0002-8604-106X
Howard Isaacson https://orcid.org/0000-0002-0531-1073
Imke de Pater https://orcid.org/0000-0002-4278-3168
Danny C. Price https://orcid.org/0000-0003-2783-1608

[11] https://github.com/bbrzycki/setigen

References

Barinova, O., Lempitsky, V., & Kholi, P. 2012, ITPAM, 34, 1773
Chollet, F. 2015, Keras, https://keras.io
Cordes, J. M., Lazio, T. J. W., & Sagan, C. 1997, ApJ, 487, 782
Drake, F. D. 1961, PhT, 14, 40
Enriquez, E., & Price, D. 2019, turboSETI: Python-based SETI search algorithm Astrophysics Source Code Library, ascl:1906.006
Enriquez, J. E., Siemion, A., Foster, G., et al. 2017, ApJ, 849, 104
Fridman, P. 2011, AcAau, 69, 777
Gray, R. H., & Mooley, K. 2017, AJ, 153, 110
Harp, G., Richards, J., Shostak, S., et al. 2019, arXiv:1902.02426
Harp, G., Richards, J., Tarter, J. C., et al. 2016, AJ, 152, 181
He, K., Zhang, X., Ren, S., & Sun, J. 2016, in Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (Piscataway, NJ: IEEE), 770
Horowitz, P., Matthews, B. S., Forster, J., et al. 1986, Icar, 67, 525
Hough, P. V. 1959, in Proc. of the 2nd Int. Conf. on High Energy Accelerators and Instrumentation (Geneva: CERN), 554
Hunter, J. D. 2007, CSE, 9, 90
Ioffe, S., & Szegedy, C. 2015, arXiv:1502.03167
Isaacson, H., Siemion, A. P. V., Marcy, G. W., et al. 2017, PASA, 129, 054501
Jonas, J. L. 2009, IEEEP, 97, 1522
Korpela, E., Werthimer, D., Anderson, D., Cobb, J., & Lebofsky, M. 2001, CSE, 3, 78
Korpela, E. J., Anderson, D. P., Bankay, R., et al. 2011, Proc. SPIE, 8152, 815212
Krizhevsky, A., Sutskever, I., & Hinton, G. E. 2012, Advances in Neural Information Processing Systems (Redhook, NY: Curran Associates), 1097
Lebofsky, M., Croft, S., Siemion, A. P. V., et al. 2019, PASP, 131, 124505
Lotova, N., Blums, D., & Vladimirskii, K. 1985, A&A, 150, 266
MacMahon, D. H. E., Price, D. C., Lebofsky, M., et al. 2018, PASP, 130, 044502
McDonough, R. N., & Whalen, A. D. 1995, Detection of Signals in Noise (San Diego, CA: Academic Press)
Monari, J., Montebugnoli, S., Orlati, A., Ferri, M., & Leone, G. 2006, AcAau, 58, 230
Nita, G. M., Gary, D. E., Liu, Z., Hurford, G. J., & White, S. M. 2007, PASP, 119, 805
Nita, G. M., Hickish, J., MacMahon, D., & Gary, D. E. 2016, JAI, 5, 1641009
O'Shea, T. J., Roy, T., & Clancy, T. C. 2018, ISTSP, 12, 168
Pinchuk, P., Margot, J.-L., Greenberg, A. H., et al. 2019, AJ, 157, 122
Prestage, R. M., Bloss, M., Brandt, J., et al. 2015, in 2015 USNC-URSI Radio Science Meeting (Joint with AP-S Symp.) (Piscataway, NJ: IEEE), 294
Price, D., Enriquez, J., Chen, Y., & Siebert, M. 2019, JOSS, 4, 1554
Price, D. C., Enriquez, J. E., Brzycki, B., et al. 2019, AJ, 159, 86
Price, D. C., Foster, G., Geyer, M., et al. 2019, MNRAS, 486, 3636
Price, D. C., MacMahon, D. H. E., Lebofsky, M., et al. 2018, PASA, 35, 41
Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. 2016, in Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (Piscataway, NJ: IEEE), 779
Ren, S., He, K., Girshick, R., & Sun, J. 2015, in Advances in Neural Information Processing Systems, ed. C. Cortes et al. (Cambridge, MA: MIT Press), 91
Rickett, B. J. 1977, ARA&A, 15, 479
Ruf, C. S., Gross, S. M., & Misra, S. 2006, ITGRS, 44, 694
Sheikh, S. Z., Siemion, A., Enriquez, J. E., et al. 2020, AJ, 160, 29
Sheikh, S. Z., Wright, J. T., Siemion, A., & Enriquez, J. E. 2019, ApJ, 884, 14
Siemion, A. P. V., Benford, J., Cheng-Jin, J., et al. 2015, POS, 116
Siemion, A. P. V., Demorest, P., Korpela, E., et al. 2013, ApJ, 767, 94
Simonyan, K., & Zisserman, A. 2014, arXiv:1409.1556
Sokolowski, M., Wayth, R. B., & Lewis, M. 2015, in 2015 IEEE Global Electromagnetic Compatibility Conf. (GEMCCON) (Piscataway, NJ: IEEE), 1

Szegedy, C., Liu, W., Jia, Y., et al. 2015, in Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (Piscatway, NJ: IEEE), 1
Tarter, J. 2001, ARA&A, 39, 511
Taylor, J. H. 1974, A&AS, 15, 367
Thompson, A. R., Moran, J. M., & Swenson, G. W. J. 2017, Interferometry and Synthesis in Radio Astronomy (3rd ed.; Berlin: Springer)
Tingay, S., Tremblay, C., & Croft, S. 2018, ApJ, 856, 31
Vogt, S. S., Radovan, M., Kibrick, R., et al. 2014, PASP, 126, 359
Weekes, T., Badran, H., Biller, S., et al. 2002, APh, 17, 221
Welch, J., Backer, D., Blitz, L., et al. 2009, IEEEP, 97, 1438
Werthimer, D., Tarter, J., & Bowyer, S. 1985, The Search for Extraterrestrial Life: Recent Developments in IAU Symp. 112 (Cambridge: Cambridge Univ. Press), 421
Worden, S. P., Drew, J., Siemion, A., et al. 2017, AcAau, 139, 98
Wright, J. T., Kanodia, S., & Lubar, E. 2018, AJ, 156, 260
Wright, S. A., Werthimer, D., Treffers, R. R., et al. 2014, Proc. SPIE, 9147, 91470J
Zhang, Y. G., Gajjar, V., Foster, G., et al. 2018, ApJ, 866, 149
Zhang, Z.-S., Werthimer, D., Zhang, T.-J., et al. 2020, ApJ, 891, 174