SETI Signal Classification: Identifying Signals Buried Deep in Noise

Akash Mahajan akashmjn@stanford.edu
Guoli Yin guoliy@stanford.edu
Marc Vaz mvaz@stanford.edu

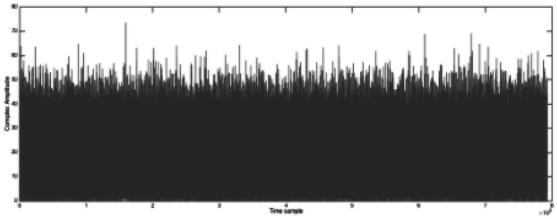
Abstract

Identifying and classifying signals of interest in an automated fashion would help SETI scale its search. In this project, we worked with a simulated dataset for the SETI-IBM code challenge, and attempted supervised classification. We developed a pre-processing pipeline for enhancing the signal and use two approaches for feature extraction/classification. For one, we improved on a previously developed algorithmic method, and for the second we designed and tuned a custom CNN architecture appropriate for this dataset. An ensemble of the two approaches obtains an overall accuracy of 80% and F-1 score of 96% at distinguishing signal from noise.

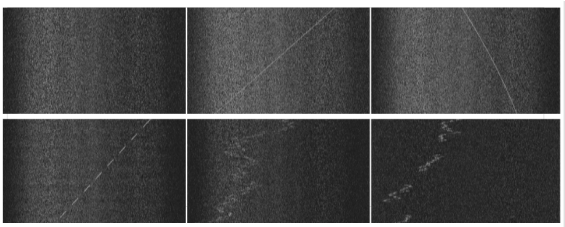
1. Introduction

1.1. Dataset

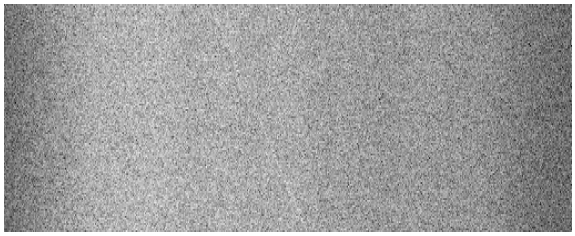
An important problem for SETI is to identify signals of interest and distinguish different signal types, some from man-made sources, from a data stream of 4TB/hour from the Allen Telescope Array (ATA). To overcome the absence of sufficient labeled data containing potential signals of interest, SETI+IBM researchers have generated a simulated dataset. The raw data consists of complex-valued time-series signals of length 792,576 as shown in Figure 1a that are to represent around 90 seconds of observation of the frequency spectrum by the ATA. This contains 5 signal types and non-stationary noise characteristics that model radiation observed from the Sun (Figure 1b). The data is more or less evenly distributed (see Table 1), however, an empirical analysis suggests that 15-20% of the data has extremely poor signal to noise ratio (SNR) with signals barely/not distinguishable from visual inspection of spectrograms (Figure 1c).



(a) Raw signal (complex amplitude vs time)



(b) Signal classes clockwise from top-left: noise, narrowband (line), narrowbanddrd (slight curve), squarepulsednarrowband, squiggle, squigglesquarepulsednarrowband.



(c) Signal with poor SNR

Figure 1: Samples from dataset

2. Methodology

2.1. Dataset division

We used a hold-out set method of cross validation and divide the dataset of 13985 signals into 10588, 2097 and 1300 training, validation and test examples. To simplify the initial stages of evaluating models, we used a smaller subset of our data comprising 4 basic classes (labels 0, 1, 2, 4) that we term *BASIC4* that consists of broad signal shapes (line,curve,squiggle) without pulses.

2.2. Approach overview

Given the raw spectrogram signal, we first converted the same into a spectrogram for analysis, by dividing the signal into time-slices and generating a periodogram for the same. For our final model, we employed a technique known as a Welch periodogram at this stage for the periodogram (Sec-

Signal Class	Count	Label
Noise	1998	0
Narrowband (lines)	1997	1
Narrowbanddrd (curves)	3995	2
Squarepulsednarrowband (pulsed line)	1998	3
Squiggle	1997	4
Squigglesquarepulsednarrowband	2000	5
Total	13985	

Table 1: Data distribution

tion 3.1) in order to improve SNR. The dimension of the generated periodogram is tunable (default provided by *ibmseti* package is 129×6144). The spectrograms were converted to images, which were used for downstream classification/feature extraction by two methods in the following steps. In Section 3.2 we used an optimization-based approach to extract a time-series of signal frequencies that is classified using manually defined features in Section 4.2. For the second approach, we developed a custom CNN architecture to classify these images (Section 4.1). In Section 4.3 we conclude with an ensemble model of the above two approaches.

2.3. Baseline Model

To establish a quick baseline, we transformed the signals to spectrograms using the default *ibmseti* package to 129×6144 spectrograms and then down-sized these to 224×1024 images (size chosen arbitrarily through visual inspection). We used the popular VGG CNN model [3] with pre-trained weights on ImageNet dataset to extract features, and classified the extracted features using a linear SVM. We found the best performance to be from the block5 layer, tabulated in Table 2. The results indicated that a major source of confusion was distinguishing signal from noise, and lines (narrowband) from curves (narrowbanddrd) possibly due to several very slightly curved signals.

Prediction		0	1	2	4
True class	0	173	1	0	4
	1	27	100	31	2
	2	74	89	212	7
	4	83	4	2	104
Overall accuracy: 0.65					
Signal v noise F1: 0.85					

Table 2: Results for baseline model on BASIC4

Prediction		0	1	2	3	4	5
True class	0	174	0	0	0	0	4
	1	29	97	25	4	1	1
	2	14	74	210	2	3	2
	3	14	7	2	82	3	16
	4	24	0	5	3	69	26
	5	17	0	1	3	59	53
Overall accuracy: 0.51							
Signal v noise F1: 0.80							

Table 3: Results for baseline model on 6 classes

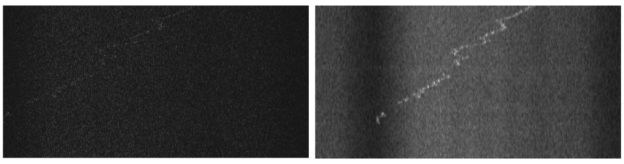


Figure 2: Image generated from FFT (left) vs Welch periodogram (right) for a squigglesquarepulsed signal (note the improved SNR)

3. Pre-processing

3.1. Signal Processing - Welch Periodogram

Each entry in the dataset consists of a time series of 792576 samples as described above. This time series can be partitioned into a number of time slices (for example 129 slices with 6144 samples). An ordinary FFT over such a slice would result in a spectrogram of 192×4128, however a Welch periodogram averages the FFT for smaller overlapping chunks of a specified size (say 256) across each slice of 6144 samples, a final dimension of 192×256. The averaging improves the SNR (Figure 2), at the cost of reduced resolution (from 4128 to 256). Using this method, we can generate spectrograms of an arbitrary dimension without loss of signal due to downsizing. Applying this method to the baseline data improves the accuracy from 0.65 to 0.71.

3.1.1 Aspect ratio / Time-frequency resolution

The impact of different aspect ratios (different time vs frequency resolution) on model performance, was studied using the BASIC4 dataset on the SVM models, using the scaling method as described above. Analysis of the confusion matrix for different aspect ratios showed that images with a larger horizontal aspect ratio were better at differentiating between the narrowband (line), and narrowbanddrd (slight curves). An ensemble model was also tried between 192×256 and 192×512 that was weighted to trust narrowbanddrd predictions more from the latter. The results

tabulated in Table 4, show the relative improvement for all methods.

192x128	192x256	192x512	Ensemble
0.66	0.71	0.70	0.72

Table 4: SVM Accuracy across image resolutions: BASIC4

3.2. Trace Extraction

For this approach, we built upon an algorithm developed by the previous year's team [1] that aims to extract a time-series of frequency values $X|x \in [1, 2, \cdots, F]$, corresponding to the frequency value of the signal at each time instant. This is formulated as an optimization problem seeking to minimize a loss that comprises a tradeoff between intensity value at a chosen point I_{x_t} and its deviation from the previously chosen point.

$$L(X) = -\sum_{t=1}^{T} \left[\alpha I_{x_t} - (1-\alpha)(x_t - x_{t-1})^2\right] \quad (1)$$

This is solved via dynamic programming, with the overlapping sub-problem being the choice of the best previous point x_{t-1} assuming that the current point x_t is chosen. As this was designed for a different dataset, we see two problems in Figure 3. For one, the algorithm is unable to catch the signal for images with poor SNR, and second, partial signals result in a set of spurious points being chosen after the signal is cut off since the algorithm is set up to generate a point for every time instant.

3.2.1 Column Normalization

An improvement to this algorithm stems from a key observation in Figure 4 (top left) where we can see that the background noise seems to have a broad 'banded' presence that is exaggerated when normalizing the image by row (top right). This characteristic was observed across several images, and is a characteristic of 'pulsars'. Normalizing by column eliminates this noise characteristic leading to a more 'even' noise and the algorithm does not get stuck at regions with high median values due to background noise (bottom right). Thus the loss is now updated, where $M_f^{x_t}$ is the median of the frequency column f corresponding to x_t.

$$L(X) = -\sum_{t=1}^{T} \left[\alpha(I_{x_t} - M_f^{x_t}) - (1-\alpha)(x_t - x_{t-1})^2\right] \quad (2)$$

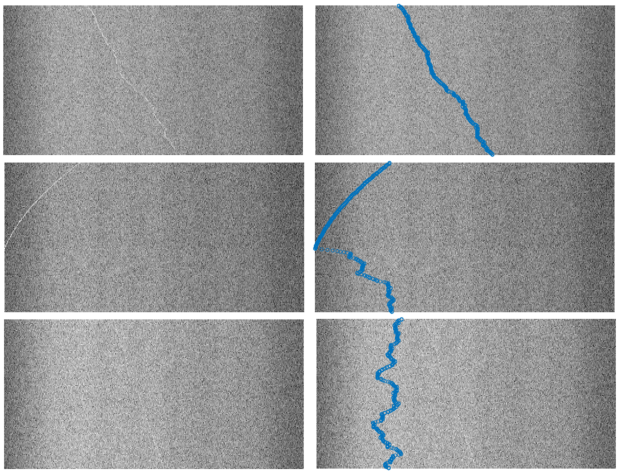


Figure 3: Trace extraction algorithm performance for different image types. (top) clear signal, (middle) partial signal, (bottom) faint signal from Figure 1c

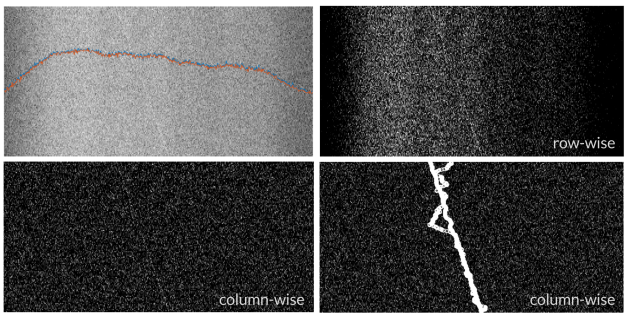


Figure 4: (top left) Poor SNR image with column-wise mean/median (orange/blue) overlayed on image, (top right) normalizing spectrogram by row, (bottom left) normalizing spectrogram by column, (bottom right) trace extraction on normalized data for $\alpha = 0.1, 0.5$

4. Classification

4.1. CNN Network architecture: SETINet

The baseline model results indicated that distinguishing unclear signals was a key source of error. This would need a model to learn better filters that are able to extract the signal from the background noise. We chose to train a custom CNN architecture for this task for two reasons. For one, the VGG architecture is large, and built for a complex problem (ImageNet consisted of 1000 classes of animals, cars, objects etc.), while distinguishing geometric signal shapes is relatively less complex once the signals are extracted. Second, the VGG architecture was trained on a drastically different dataset, while this problem needs fine-grained filtering to bring out faint signals from noisy data. Attempting to fine-tune the filters of a large architecture for a very differ-

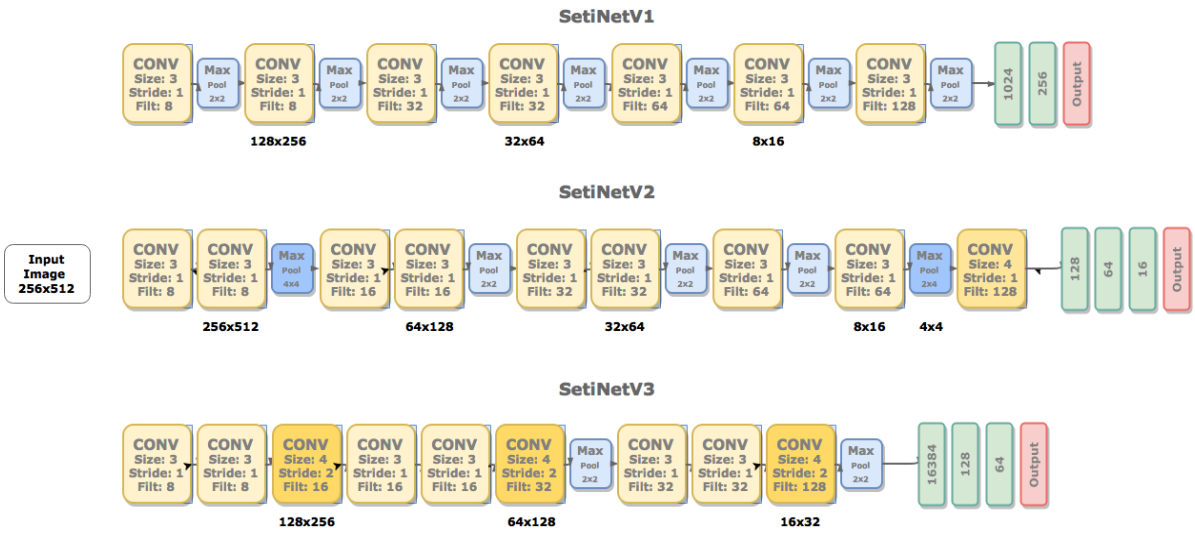


Figure 5: Comparison of different CNN model architectures

ent problem would be ill-advised, unless our dataset was of similar scale. For initial development, we use the BASIC4 dataset. A comparison of architecture iterations is shown in Figure 5.

4.1.1 V1 Architecture

As a starting point, we began with an architecture, loosely based on the LeNet[2] architecture used for MNIST digit recognition, having a relatively small number of filters as signal geometries once extracted, were a similar problem to distinguishing digits. We observed relatively poor performance from the model, which was overfitting despite an extensive search of hyperparameters, yielding an **overall accuracy 0.51** on BASIC4, which was worse than the baseline. A closer look at the model activations showed that 4-5 Convolution+ReLU layers were needed to begin to denoise the signals, at which point our image sizes were too small, and shapes would be difficult to discern given only 7 such units in this model.

4.1.2 V2 Architecture

The next iteration of the model used a total of 9 Convolution+ReLU layers, and were grouped into 'blocks', before a max-pooling operation, as from SETINet V1 we observed that multiple Conv layers were needed to extract the signal. This resulted in a smaller number of 5 maxpooling blocks (compared to 7 earlier). We attempted to keep the final activation size before the final fully connected network small as we had observed V1 to be overfitting. Hence this resulted in more aggressive maxpooling at the beginning and at the end as in Figure 5

To prevent the model from overfitting, we performed data augmentation, such as horizontal flips, vertical flips and minor ($\leq 10\%$) vertical/horizontal crops to multiply our training data by $\times 3$. Also, since training a single model was proving to be difficult, we trained **two separate models**: one for signal vs noise, and one to distinguish between 3 classes. This gave a slightly improved **overall accuracy of 0.66** and a **signal v noise F1 of 0.905**.

4.1.3 V3 Architecture

Examining the activations of SETINet V2, we saw that extracting poor SNR images was still happening towards the end of block 3. The activations size at that point was greatly reduced to 32×64 which seemed too small to capture subtle curves and differences amongst signals that are only a few pixels wide (see Section 7 Appendix). In the V3 architecture, we avoided the use of max-pooling layers, instead using Convolution layers of size 4×4 with a stride of 2 to downsize as in Figure 5. This resulted in a larger activation of 64×128 after 6 purely Conv+ReLU blocks (see Section 7 Appendix). This yielded a much improved **overall accuracy of 0.78** and a **signal v noise F1 of 0.945**. Performance is compared in Table 5.
With a promising architecture, we used the full dataset (6 classes), using Welch spectrograms (Section 3.1) to generate images of our size 256×512 without any downsizing loss, and column normalization (Section 3.2.1). Using the same approach as in V2, we trained 2 separate models and ensembled their predictions for a large improvement on the baseline (Table 6). We verified that the ensemble of 2 separate models performed slightly better than a single model that obtained accuracy of 0.78 compared to 0.79.

	BASIC4 class accuracy	Noise vs Signal F1
Baseline	0.65	0.85
SetiNetV1	0.51	0.67
SetiNetV2	0.66	0.905
SetiNetV3	0.78	0.945

Table 5: SETINet performance comparison on BASIC4

Prediction		0	1	2	3	4	5
True class	0	172	0	3	0	9	1
	1	10	**146**	**27**	0	2	0
	2	13	**50**	**277**	1	7	1
	3	12	8	8	134	2	28
	4	20	0	10	0	134	22
	5	14	0	1	4	24	160
Overall accuracy: 0.79 Signal v noise F1: 0.962							

Table 6: Results for SETINetV3 ensemble on full dataset

4.2. Trace feature extraction

For this approach, we use as data the time-series vectors X and I_{x_t} from Section 3.2 that are the chosen frequencies and intensities at the corresponding frequencies (Figure 6). From these, we extract the following features, for two different values of $\alpha = 0.1, 0.5$ (constrained, more 'squiggly'):

1. Order 2 Polynomial fit for X (2×3 features) - Since we are looking to distinguish lines/curves (that are conics/parabolas), precise values should help distinguish slight curvature. As seen in Figure 6, we estimate these using RANSAC regression as opposed to ordinary least squares, to only fit the set of inlier points in cases where the signal is partially cutoff.

2. Mean Squared Error of fit for X (2×1 features) - Should help differentiate geometric signals such as lines/curves from others.

3. Total variation, Kurtosis for X, I_{x_t} (2×4 features) - Variation is measured as std[diff(X^*)], where X^* is a rolling mean smoothed series of X to reduce the noise before differencing. The Kurtosis helps differentiate purely noisy signals (closer to Gaussian) from others.

4. Mean Squared difference between X for two different values of $\alpha = 0.1, 0.5$ (1 feature) - For signals that are purely noise, we expect less consistency between the traces for two extreme values of α.

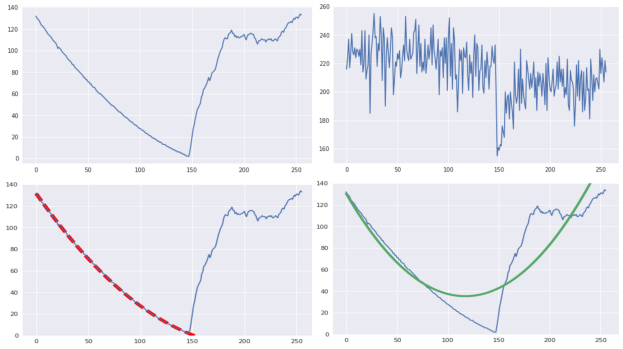


Figure 6: (top left) X and (top right) I_{x_t} extracted from cutoff signal in Figure 3, (bottom left) polynomial fit via RANSAC, (bottom right) polynomial fit for least squares

As the number of features was significantly smaller than our baseline, we could build a kernel-SVM model with a gaussian kernel which yielded better performance than a simple linear SVM. Results for this model are in Table 7, which is an improvement on the baseline. While it is worse that the SETINet V3 model at distinguishing signal from noise, it is slightly better at distinguishing classes 1 and 2 (narrowband, narrowbanddrd).

Prediction		0	1	2	3	4	5
True class	0	162	0	2	0	10	11
	1	15	**144**	**0**	20	4	2
	2	44	**32**	**247**	15	6	5
	3	50	11	3	111	5	12
	4	72	1	4	1	72	36
	5	112	1	2	3	34	51
Overall accuracy: 0.61 Signal v noise F1: 0.84							

Table 7: Classification results for trace-extraction model ($C = 100, \gamma = 1/17$)

4.3. Ensemble Model

While the trace-extraction model was able to obtain better performance for classes 1 & 2 that were confused by the CNN model, the overall accuracy for the CNN model was significantly higher. We attempted to combine the strength of both models by building an ensemble model. To do so, we concatenated the 17 features described in Section 4.2 with the softmax outputs of our 2 separate CNN models (signal v noise and 5 signal classification) in the form of

Prediction		0	1	2	3	4	5
True class	0	175	2	2	1	4	1
	1	9	**162**	**7**	4	1	2
	2	15	**50**	**278**	1	4	1
	3	14	11	3	141	1	22
	4	24	1	5	0	132	24
	5	17	0	1	10	20	155
Overall accuracy: 0.80 Signal v noise F1: 0.955							

Table 8: Results for SETINetV3 ensemble on full dataset ($C = 0.8, \gamma = 1/24$)

class probabilities (2 + 5 features). As earlier, we trained a kernel SVM with a gaussian kernel on these features which resulted in a clear improvement on classes 1 and 2 and slight improvement overall (Table 8).

5. Conclusion & Further Work

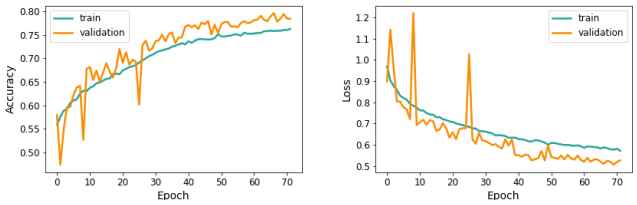


Figure 7: Loss curves for SETINetV3 trained on full dataset

In conclusion, we were able to achieve a good accuracy on the dataset, as manual error analysis of misclassifications from our best model shows signals that are highly buried in noise, most of which would not be identifiable by human eye. By ensembling our two approaches we were able to combine the strengths of the two models. While our final CNN model architecture showed a great improvement, some further work we think would lead to further improvements is listed below.

1. Improve manually generated features to help distinguish classes 4 and 5 (squiggle, and squigglesquarepulsed).

2. Use techniques such as hard negative mining for further training of the CNN model to improve performance on difficult classes.

3. Extend the trace extraction algorithm to make an additional decision of whether to include a point, to deal better with partially cutoff signals.

4. As seen in Figure 7, we can see that for the best model, the training performance is not significantly better than validation. While this is good generalization, it might suggest room for a larger model until there is significant overfitting. Based on our work so far, a likely option could be a modification on the V3 model with no downsizing in block1 and extra conv layers downstream.

5. Extend the approach in 3.1.1 to use an ensemble of CNN models looking at different image aspect ratios.

6. Acknowledgements

We would like to thank Prof. Jeffrey Ullman and Andreas Paepcke for their regular feedback and mentorship. We would also like to thank Adam Cox and Graham Mackintosh from the IBM-SETI team for their feedback and assistance while working with the dataset. Lastly, we would like to thank Justin Johnson and Serena Yeung from the CS231n class for feedback while designing our models.

References

[1] J. W. Frank Fan, Kenny Smith. Project seti: Machine recognition of squiggles in seti signal data, 2016.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

7. Appendix

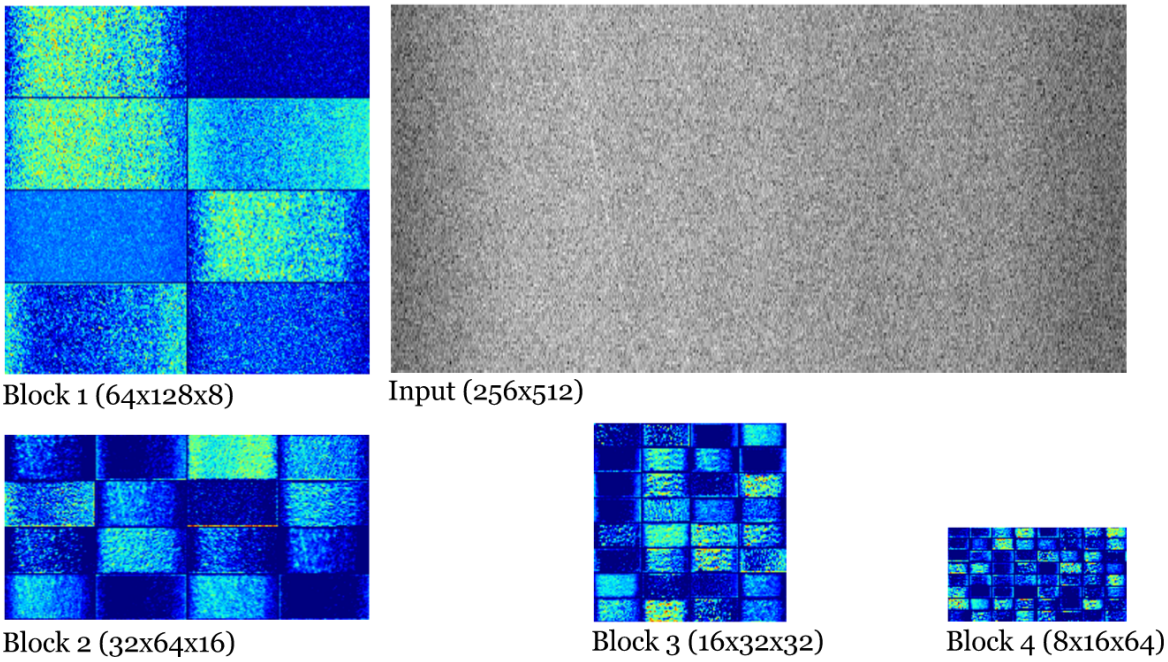


Figure 8: Visualizing SETINetV2 activations, to scale relative to input image. Each block corresponds to the next max-pool layer (blue in Fig 5). For a low SNR image, signal is extracted by block 2/3 by when the activation size is quite small.

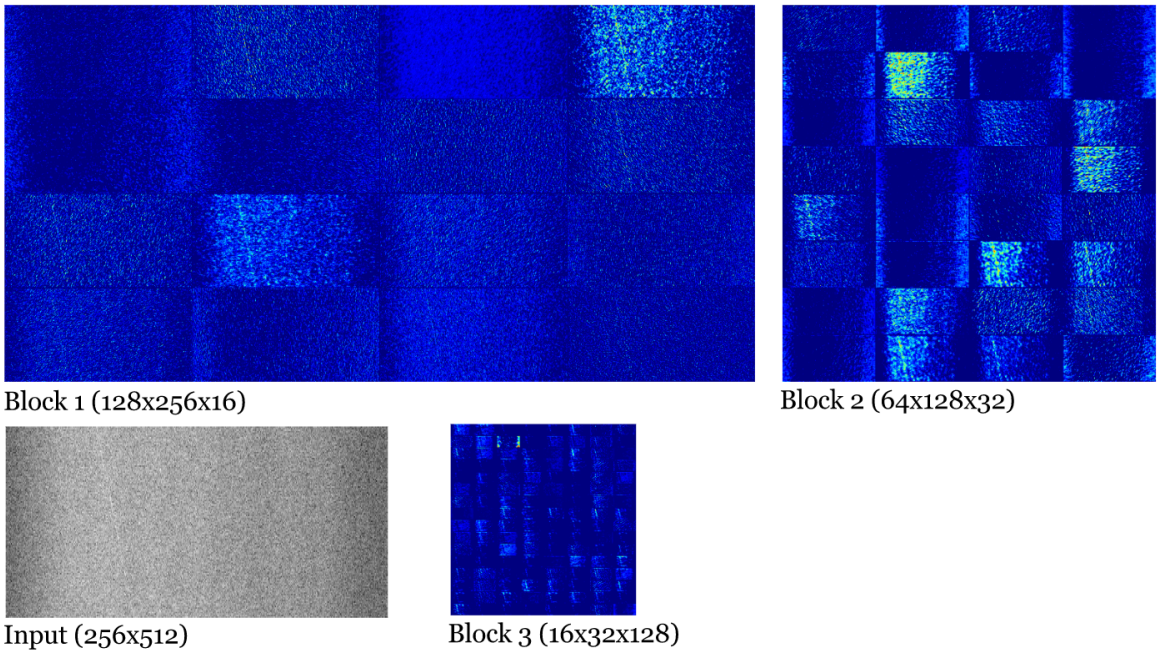


Figure 9: Visualizing SETINetV3 activations, to scale relative to input image (note the larger activation sizes). Signal is extracted at block 1/2 enabling next layers to distinguish shapes of signals that are a few pixels wide in the original image.