Using OmniSIG SDK to Create a Drone Detection Model

DEEPSIG

Using OmniSIG SDK

to Create a Drone Detection Model

Abstract

This application note details the process of using the OmniSIG SDK to create a custom neural network model for OmniSIG that is focused on performing signal classification on wireless signals emitted from small commercial drones. This note will go through the end-to-end process of capturing data to create the dataset, to labeling and annotating the data, training the new model using this new dataset, and finally deploying the model as an OmniSIG runtime sensor.

Contents

Introduction to Commercial Drone Signals

Commercial and hobbyist drones come in many different shapes and sizes, both physically and with respect to their wireless control and data signal types. Nikola Tesla had one of the first visions of unmanned aerial vehicles and described his vision in the patent, "Method of and apparatus for controlling mechanism of moving vessels or vehicles," that was granted November 8, 1898. In this patent he suggested that these vehicles would be controlled by "waves, impulses, or radiations".

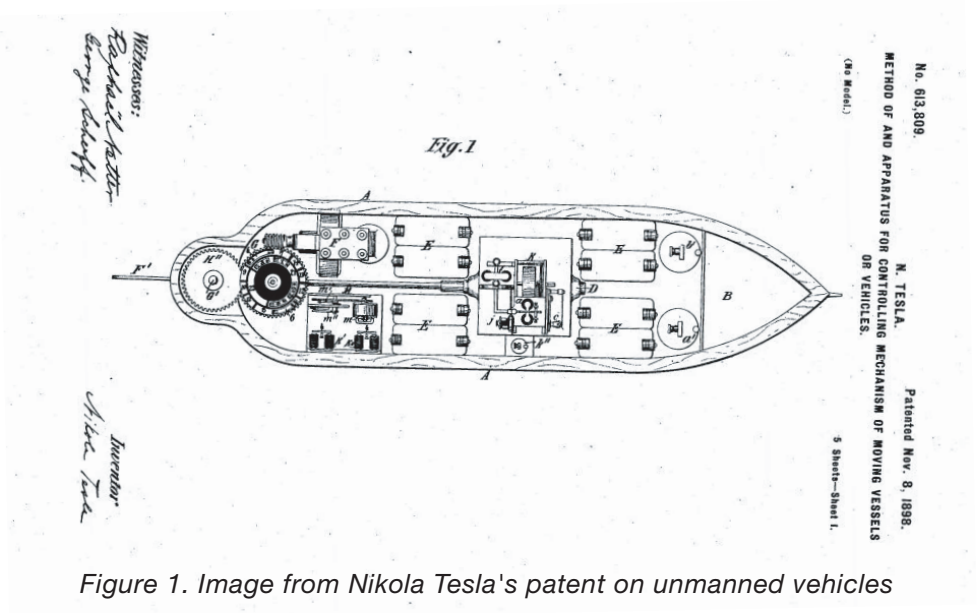

Figure 1. Image from Nikola Tesla's patent on unmanned vehicles

It was not until 2006 that non-military, commercially available drones really started "taking off". The US Federal Aviation Administration (FAA) issued its first commercial drone permit, and for the next 8 years they issued about two permits a year. It was not until Amazon CEO, Jeff Bezos announced in 2013, that the company would explore using drones as a delivery method that the commercial drone industry really began to get big. In 2015 the FAA issued over 1000 permits, which grew to 3100 in 2016. Today, hobby drones have grown into a multi-billion-dollar industry with over three million personal drones estimated to have been manufactured in 2017 alone.

The drone startups have been concentrated in the US, China, and Israel, with Dajiang Innovations (DJI), in China, accounting for 36% of the North American commercial drone sales in 2019.

Commercial drones are controlled and send data, primarily video, back over wireless connections. The types of wireless connections that the drones use can vary. Some use proprietary protocols while others use more off-the-shelf mechanisms. Many commercial drones, such as those from early versions of DJI and Parrot, used 802.11 or WiFi, in the 2.4 GHz frequency band, as the technology for transmitted data. Many inexpensive commercial drones still use WiFi for control today. WiFi is convenient for manufacturers as commercial chips are cheap, readily available and allow consumers to control and receive data easily from their smartphone. However, WiFi does have disadvantages. It is inherently limited in range and when flying in areas with other WiFi networks the signal quality may be degraded. This decreases the range even further or causes poor video quality. In recent years manufacturers have been adding in their own transmission systems that provide for better interference mitigation and longer ranges. This complicates the development for creating systems that detect and classify different types of drone signals as the drone wireless communication systems ecosystem has become much more diverse each day.

Wi-Fi-based Systems

Control and data packets embedded in standard 802.11 frames on either the 2.4 GHz or 5.8 GHz band. In many cases the drones will automatically switch between the 2.4 GHz and 5.8 GHz bands based on the current wireless environment to avoid interference.

Common drones that use this transmission system:

Parrot Bebop 1, Parrot Bebop 2, DBPower UDI, DBPower Discovery, DJI Tello, Tenergy TDR, Wingsland, DJI Spark (without the DJI controller), Mavic Air (without the DJI controller)

Wi-Fi-based systems can easily be detected using standard 802.11 discovery mechanisms. The manufacturers are allotted blocks of MAC addresses which allow Wi-Fi survey systems, such as Kismet, to quickly discern whether a Wi-Fi enabled drone is in the local area. Most also have well-known wireless network identifiers (SSID's) that can be detected.

Detecting and classifying signals for drones that do not use Wi-Fi for their control and data is traditionally much more difficult as many of the signal types are proprietary and can operate on a wide range of frequencies.

Non-WiFi Transmission Systems

There are thousands of drone companies that exist today with products on the market. Some are Wi-Fi and many are not. As an example, DJI has three separate proprietary transmission systems for their drone products alone. These protocols are less likely to be affected by interference and have a longer transmission range.

Common drones that use DJI's Ocusync and Lightbridge (LB / LB2) protocols:

DJI Mavic Pro (Ocusync), Phantom 4 Pro V2.0 (Ocusync), Phantom 4 Pro (LB), Phantom 4 Advanced (LB), Inspire 2 (LB2), Matrice 200 Series (LB2) and Matrice 600 Pro (LB2)

The hobbyist / drone sport market also has a number of other general purpose controllers that use the ISM bands (915 MHz, 2.4 GHz, and the 5.8 GHz bands) with complex frequency-hopping controllers such as those made by FrSky and FlySky. There are also long-range frequency-hopping telemetry systems that support the MAVLink protocol which operate in the 433 or 915 MHz ISM bands.

For these systems, as opposed to WiFi-based systems, the uplink, or the control signal from the controller to the drone is a completely different signal type than the video feed coming from the drone to the operator. It is possible that hobbyist drones can in fact use frequency-hopping controllers, MAVLink telemetry, and WiFi video all on the same platform.

In terms of signal characteristics, they range from very wideband OFDM signals, that typically carry data from the drone down to the controller, to small narrowband bursts that hop around the spectrum. These latter hopping transmissions are typically the control signals which need to be more robust to interference so as not to create a situation where the drone is not controllable.

Figure 2. DragonLink Drone Range Extender

Range extension systems, like the one shown in Figure 2, are also available for commercial drones that commonly operate at 433 MHz. This lower frequency provides a much greater range but does not allow enough bandwidth to provide a high-quality video link. There are many different systems that operate in the 433 MHz band, each having their own signal type. These signal types are typically some variant of a narrowband FSK signal.

Common range extension systems:

DragonLink, EzUHF, OpenLRS

The OmniSIG sensor and the OmniSIG SDK are well suited for building systems that quickly detect and classify commercial drone signals across all of these areas. Whether they are 802.11-based and OmniSIG is paired with a WiFi post-processor or they are not 802.11-based and the OmniSIG SDK is used to train a neural network to recognize the signals, OmniSIG provides a innovative way to detect and classify commercial drone signals at low SNRs and the SDK provides a way to add new drone signals to the system within hours.

OmniSIG provides a innovative way to detect and classify commercial drone signals at low SNRs and the SDK provides a way to add new drone signals to the system within hours.

The next sections will overview the SDK and walk through an example of using the OmniSIG SDK to build an RF dataset that consists of DJI Ocusync uplink and downlink signals. We will show how this dataset is used by the SDK to train a new OmniSIG model that can be used by the runtime OmniSIG sensor to immediately detect and classify DJI Ocusync signals.

OmniSIG Sensor and SDK Overview

OmniSIG provides a new class of RF sensing using DeepSig's pioneering application of Artificial Intelligence (AI) to radio systems. Going beyond the capabilities of existing spectrum monitoring solutions, OmniSIG's custom deep learning approach leverages convolutional neural networks in addition to several custom RF tailored network architectures that take advantage of complex baseband IQ data using both the time and frequency domains, to maximize the features learned by the AI.

Compared to traditional methods, OmniSIG's approach provides higher sensitivity and is more robust in harsh and dynamic spectrum environments, while also requiring less dynamic range and computational resources. For drone detection applications, OmniSIG can achieve increased sensitivity for detecting a wide range of drone signal types, effectively allowing the sensor to have an increased standoff. The sensor component can be integrated onto a variety of hardware platforms, both embedded and large. It has been developed with an open API and standard output specifications that allow it to be easily integrated into external systems.

DeepSig provides a default network model for the runtime sensor that includes support for detecting and classifying a variety of signals. A powerful capability comes into play with the OmniSIG SDK, where users can create their own network models using their own datasets to enable the sensor to detect and classify the customers specific signals-of-interest.

SIGNAL TYPES

OmniSIG's default model supports the detection and classification of the following signal types:

- LTE
- WCDMA
- CDMA2K
- GSM
- P25
- FM
- WIFI
- BLUETOOTH
- ATSC
- DMR
- LTE UPLINK
- DCS/DPL

OmniSIG also identifies unknown signals that it unable to classify as a trained signal type. This is useful for immediately identifying anomalies.

Additional signal types can be added by using the OmniSIG SDK tool.

The OmniSIG SDK contains tools for:

1) Sorting, labeling, and curating RF data

2) Training an OmniSIG Sensor model with labeled data

3) Evaluating its performance

4) Deploying the trained deep learning model into an OmniSIG runtime

This tool suite is a market-first, enabling customers to custom-tune DeepSig's deep learning models for signal detection and classification for their specific RF signatures and applications.

OmniSIG SDK was designed by engineers with decades of industry experience to enable signal processing on complex-valued RF sample data. It contains specialized features to assist in working with large RF datasets that don't exist elsewhere.

The OmniSIG sensor and the OmniSIG SDK combine to make the full capability development process. The standard process begins with the building of the training set. In the case of the drone data set, this consists of taking over-the-air snapshots of drones in a variety of operating modes. These snapshots are then labeled using the OmniSIG SDK and provided to the training routines that create a specific OmniSIG neural network model file that is trained to identify the signals that were annotated. The model file is passed to the OmniSIG engine when the OmniSIG sensor is started and the new signals can now be detected and identified in real-time. This overall process is depicted in Figure 3.

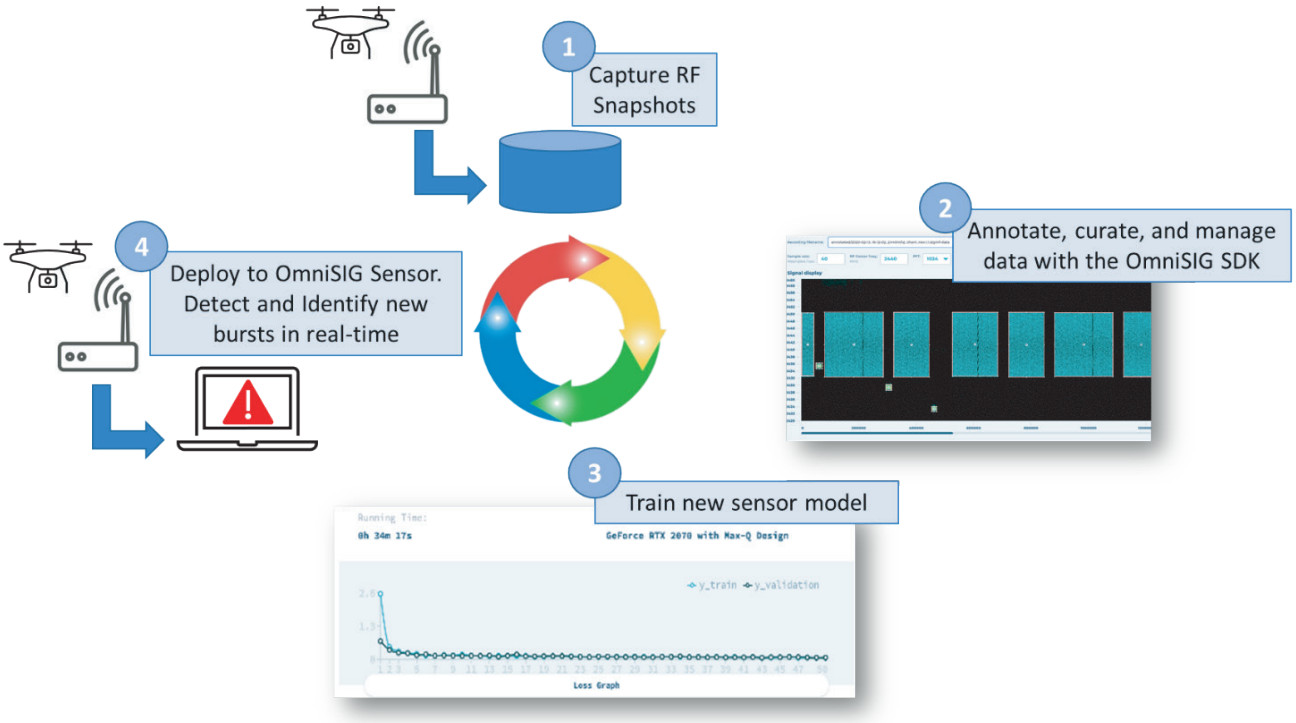

Figure 3. OmniSIG capability development workflow

The following sections detail the specifics of this process for developing a new custom neural network model, using the OmniSIG SDK, to detect and classify several different types of commercial drone wireless signals.

Building the Training Dataset

The first step in training OmniSIG to recognize a new signal is to capture a few snapshots using a software-defined radio (SDR). For this exercise, we will be working with a DJI Mavic Pro drone. If you do not have access to a particular model of drone, it is possible to use captures from the field, but it may require additional care in the selection of radio hardware and cleaning of training data.

Capturing RF snapshots can be done in a variety of ways using a variety of different hardware and software platforms. In addition, the diagnostic user interface for OmniSIG has a record feature that is capable of recording complex 32-bit float IQ data streams. OmniSIG can work with multiple types of data formats including complex 16-bit integer and complex 32-bit floats (these two are the most common). The training dataset can consist of RF recordings from different radios and still maintain a high-performance classification. DeepSig leverages radios from many different venders for testing and demonstration purposes, including Epiq Solutions, National Instruments, Herrick Technologies, and many more.

For creating the Ocusync dataset, we connected a software-defined radio to a gaming laptop, with an Nvidia RTX2070 GPU and verified that the radio was operating properly. Since this model of drone operates in the 2.4 GHz and 5 GHz ISM frequency bands, it is helpful to move to a relatively RF isolated environment and power down any electronic devices using Wi-Fi or Bluetooth including phones, tablets, and Wi-Fi/Bluetooth radios on the recording laptop. Building the dataset in an isolated environment helps when labelling the data as it helps minimize unwanted signals. It is also recommended that if the drone configuration supports manual or fixed channel configuration, that this mode be enabled. This will lock the drone and controller onto a single frequency band that is easily identified, and will prevent the drone from reconfiguring its frequency bands while recording is happening (just remember to change it back to auto later if the mode is changed for signal capture).

In our recording case, the drone was configured to use a manual channel setting in the middle of the 2.4 GHz ISM band. Figure 4 shows the OmniSIG diagnostic interface with a live drone signal active as seen by our SDR with an antenna.

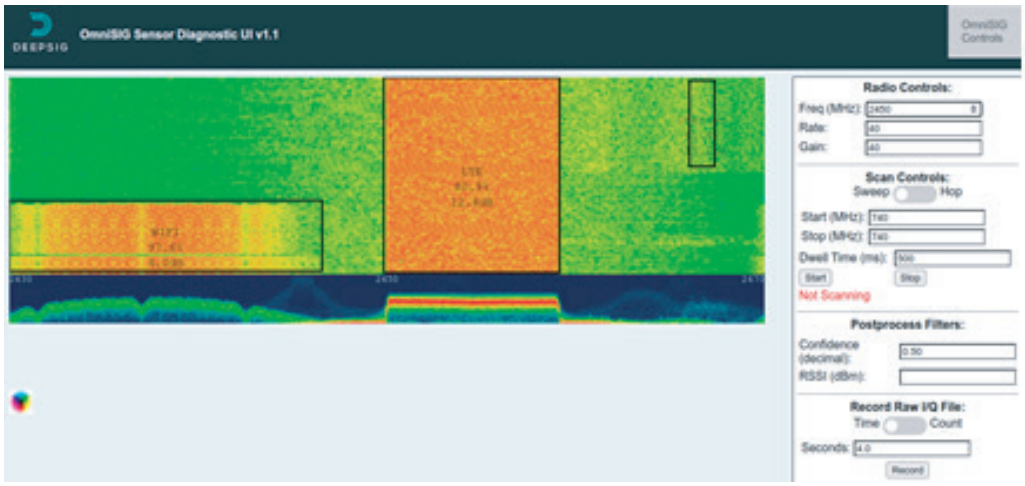

Figure 4. Acquiring an RF capture using the OmniSIG engineering interface

Since we have not trained a model yet to specifically classify it as the Ocusync protocol, the default model does its best to classify the signal. Since Ocusync is similar in many ways to LTE, in the screenshot above, you can see that the model is classifying it as LTE. OmniSIG is reporting its confidence level as ~80%. Typically, anything below 90% would indicate that this may be a different signal type than previously trained, but it is similar in signal features to the trained signal.

On the right-hand side of the diagnostic interface, if you click the OmniSIG controls button at the top, a set of controls will appear. Within this section will be several options and a button to trigger a recording. These recordings can be based on time (the default), or number of samples.

Our isolated training snapshots were recorded in the stairwell of our office building in the following manner.

1) Start the OmniSIG Sensor and navigate to the diagnostic GUI at http://localhost:4000
2) Open the controls (upper right button) and tune the radio to find the drone uplink and downlink signals
3) Adjust the gain so that the signal is clear but not clipping, which may be lower than expected if the source is nearby
4) Set the recorder to a duration of 1 second
5) Record 5-10 snapshots, with at least a few extras that can be used if some of the others are contaminated with too many interfering signals

While we used the OmniSIG sensor interface itself to collect the signals, any method for recording RF snapshots will suffice as long as the data gets written to disk in either complex 16-bit integer or 32-bit float formats.

We recorded a total of 15 seconds of Ocusync transmissions from the drone and controller and selected the best 5 seconds to use for training. Reasons for discarding data included signal clipping, due to high gain, and interference from unknown emitters. This process could be further enhanced by taking the drone to an area outside where it could be flown in a real-world environment, and additional captures could be taken while real in-flight control signals are also present.

Labelling and Annotating the Data

After the RF snapshots have been collected, they must be properly labeled so that the training process can properly teach the neural network about the signals. The next step is to get the RF snapshots that have been collected into the OmniSIG SDK filestore. The user can simply drag and drop the RF snapshots from the drone into the web-based SDK filestore allowing access to those files for the rest of the SDK functions.

The next step is to annotate or literally mark up the RF snapshots to indicate which bursts of energy correspond to the drone uplink and downlink signals. This is a good time to screen the snapshot for any stray signals and discard or crop the snapshot accordingly.

Quick Trick to Speed up Annotation

Once the first snapshot has been annotated train a model using only that first snapshot. Run additional RF snapshots through OmniSIG using that model. The output files are then imported into the SDK and typically only require minor adjustment to accurately label the signals.

Figure 5 shows the OcuSync RF snapshot loaded into the OmniSIG SDK labeling tool, ready to be labeled. The high-bandwidth OcuSync downlink signal of the Mavic Pro is composed of 10 or 20 MHz wide OFDM bursts. The uplink signal is a frequency hopper that is transmitted during gaps in the downlink. We have annotated these separately with the “OcuSync Control” and “OcuSync Video” labels

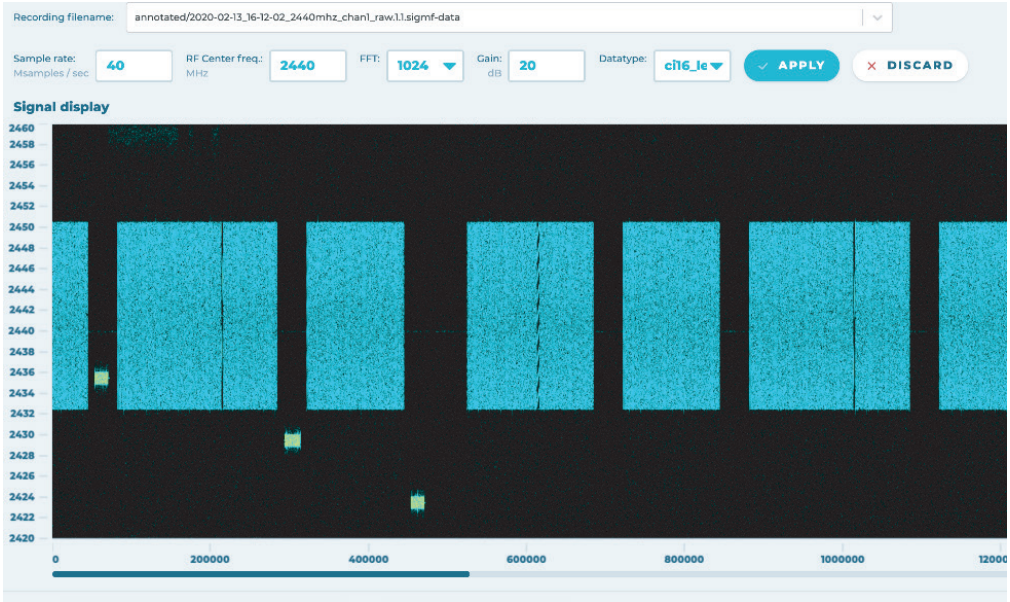

Figure 5. Fresh RF snapshot loaded into the SDK annotation tool

Figure 6 shows the RF snapshot from the Ocusync dataset after labelling is complete. Both the downlink signals and the uplink signals have been annotated, each labelled with their own specific signal type.

Annotating each file by drawing boxes around the active signal regions and assigning a label is straightforward, and each 1 second snapshot takes about 15-20 minutes to annotate by an experienced user. This is because a 1 second snapshot may consist of hundreds of bursts that need to be annotated. Once the annotations are complete, the metadata file that contains the annotated data can be saved. It will save this file in the same location as the actual RF snapshot.

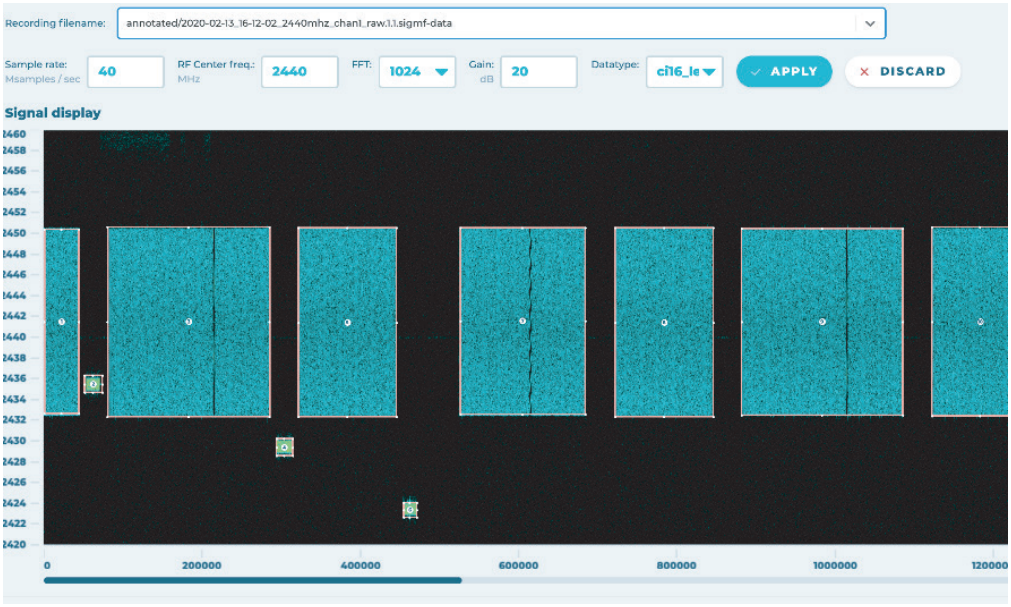

Figure 6. OmniSIG SDK fully annotated Ocusync control and data signals

To assist users in the annotation process, the OmniSIG SDK also has an “auto-annotation” feature. This feature provides a quick triage of the data and tries to draw boxes around the energy in an automated fashion. The intent is to triage the data so that the user only needs to annotate at most 20-30% of the data and more time can be spent in the training process as opposed to the labeling and annotation process.

Training the New Custom Model

Once annotation is complete, for all the data of the signals of interest, in this case, the Ocusync RF snapshots, we select the files that we would like to include into this new custom model on the “Training” tab. At this point, training commences at the click of the “Train” button. Figure 7 shows a snapshot of the training page while training the Ocusync models. Users are presented with a basic loss curve that indicates how well the training process is proceeding. The lower the curve, the less error there is between the current set of trained weights and the automated test set that is created during the training process. The SDK will stop training once the loss curve converges or the user can manually stop the training.

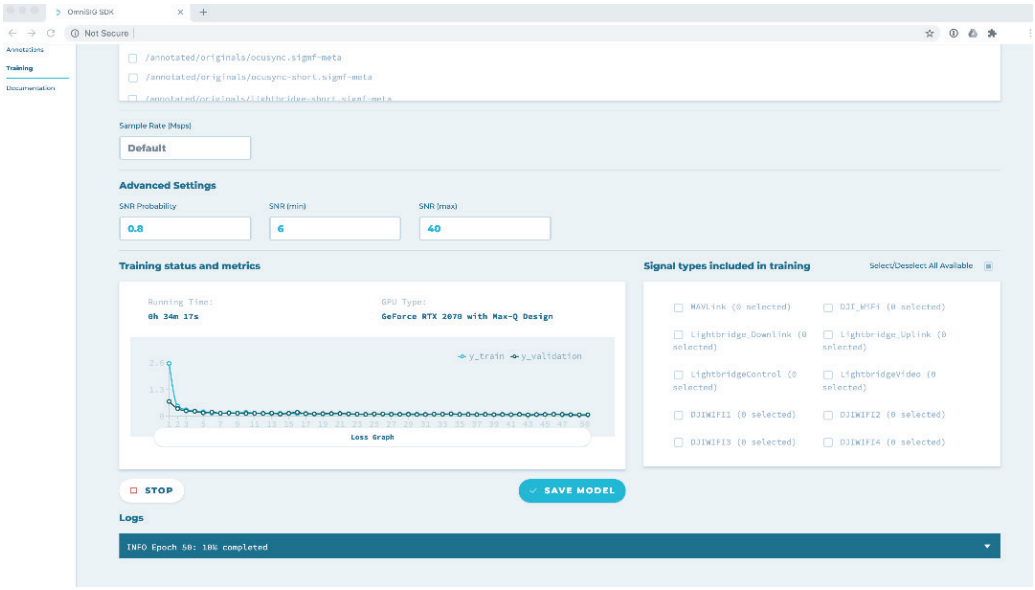

Figure 7. OmniSIG SDK 50 epochs into training Ocusync custom model

Training the drone detection model takes about six to eight hours on a high-end gaming PC, using both the annotated drone snapshots and the available DeepSig Training Dataset, which adds to the robustness of the model in real-world scenarios and minimizes false detections on similar OFDM signals. If you want a custom model that will only identify the annotated drone snapshots, training on the same PC would take approximately 1-2 hours. Other signals would either not be identified or would be identified as “Unknown” signal types.

Training can be performed on a variety of hardware platforms. For this whitepaper, the Ocusync custom model was trained on a laptop with an Nvidia RTX2070 GPU and 8 GB of video memory. This configuration is the minimum requirements that we would suggest for training custom OmniSIG models.

Deploying the Model in the OmniSIG Sensor

Deploying the new custom model to the OmniSIG sensor is remarkably simple. Once the new custom model is saved, simply pass the location of the model file in as a command line parameter when starting the OmniSIG sensor. This causes the neural network within OmniSIG to use the parameters and labels of that custom model as opposed to the default model that is built into OmniSIG.

The custom model file size ranges from 5 – 20 MB in size and can be easily pushed over a low-bandwidth networks to a remote OmniSIG sensor to immediately enable new capabilities.

The OmniSIG Sensor diagnostic GUI can be used to verify that the trained model is operating correctly. Powering on the drone and tuning the radio to its uplink and downlink frequency should display detections equivalent to the labels that the snapshots were annotated with, in this case “OcuSync Video” and “OcuSync Control”

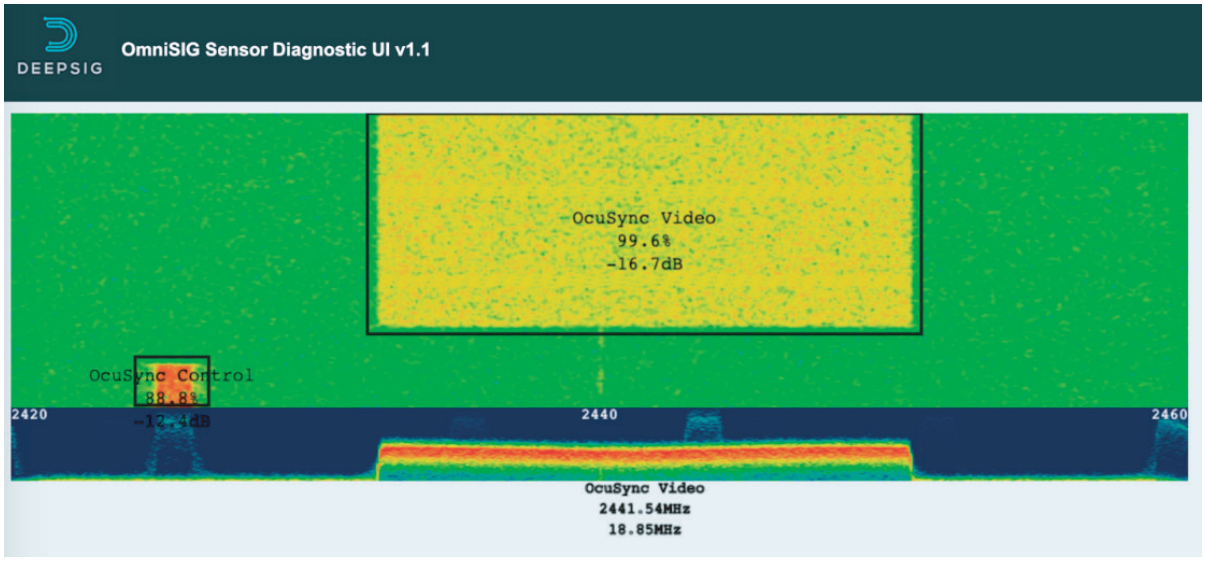

Figure 8. After training, the sensor can immediately detect and classify the Ocusync signals

With this new custom model trained, users can now view the engineering GUI, as shown in Figure 8, or output a stream of classifications to an external post-processor for further action. OmniSIG can publish the annotations several different ways, including ZeroMQ, websockets, over the network to an ElasticSearch database for analysis with the Kibana data visualization tools (the ELK software stack), or simply output to files.

Figure 9 shows one example of the dashboard elements that can be created using the OmniSIG Sensor along with the ELK stack. It is straightforward to create powerful spatial, temporal, or spectral visualizations that allow you to explore the RF environment around identify radio anomalies.

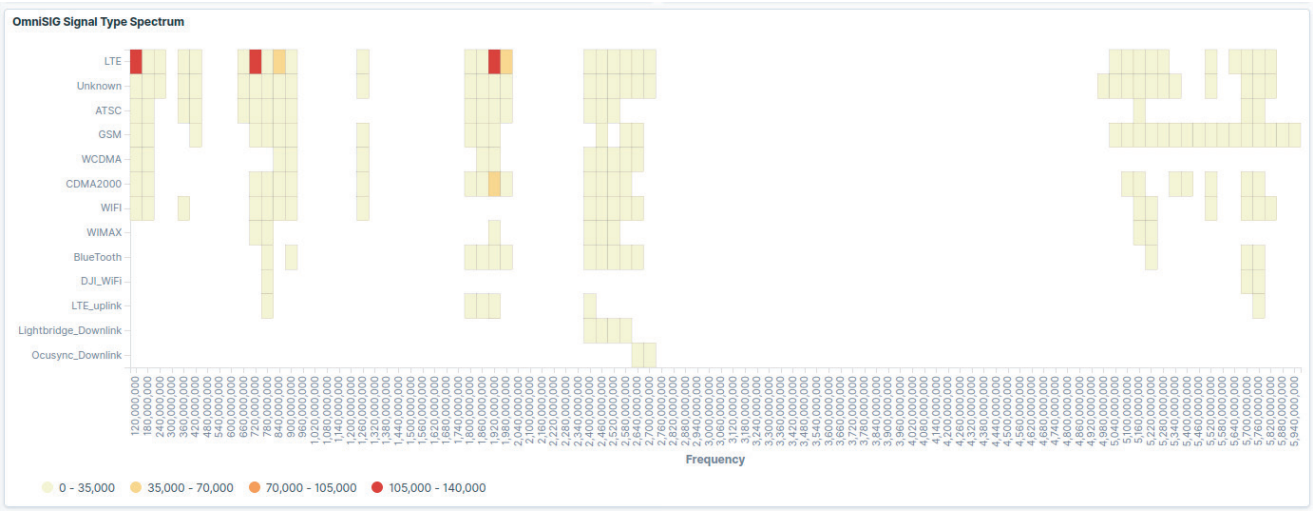

Figure 9. An example Kibana dashboard element to view the distribution of detected signal types in the scanned radio spectrum

Integrating the OmniSIG sensor into any system, to act as the flexible front-end RF survey component, is straightforward as it has been developed using open specifications and users are provided with well documented API's for both the input, control, and output of the sensor. The OmniSIG sensor will increase the sensing and scanning speed of systems, while also adding in new ways to continually evolve the capabilities of the system. Whether its drone signals, commercial cellular signals, or some other unknown signal type, the OmniSIG sensor can classify it by training on a very little amount of data.

For more information about our drone datasets and using the OmnniSIG sensor, please see our website at DeepSig.ai or contact us at info@deepsig.ai.