# Radio Machine Learning Dataset Generation with GNU Radio

**Tim O'Shea** OSHEA@VT.EDU
Bradley Department of Electrical and Computer Engineering, Virginia Tech, Arlington, VA

**Nathan West** NATHAN.WEST@OKSTATE.EDU
School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK

## Abstract

This paper surveys emerging applications of Machine Learning (ML) to the Radio Signal Processing domain. Provides some brief background on enabling methods and discusses some of the potential advancements for the field. It discusses the critical importance of good datasets for model learning, testing, and evaluation and introduces several public open source synthetic datasets for various radio machine learning tasks. These are intended to provide a robust common baselines for those working in the field and to provide a benchmark measure against which many techniques can be rapidly evaluated and compared.

## 1. Motivation & Background

Machine learning has advanced very rapidly in the past 10 years and there are several key reasons for this. Algorithms have improved dramatically in many ways including momentum methods of gradient descent and improvements in regularization and dropout, among other things. Computational power and concurrent programming models to fully leverage it have also improved dramatically, due largely to high level languages which compile efficiently to concurrent GPU implementations. Access to large and well curated datasets is another key enabler which has allowed people to rapidly compare ideas, evaluate new approaches, and train models with large parameter spaces.

Researchers in computer vision, voice recognition, natural language processing, medical imagery and finance currently push many of the state of the art advances in machine learning. Unfortunately radio signal processing has been notably absent from much of this recent work, but the potential for applications of recent advances in machine learning to the radio domain is enormous right now. Communications researchers are now increasingly considering these methods, but lack common benchmarks and open datasets for evaluating advances as seen in other application areas.

*Proceedings of the $6^{th}$ GNU Radio Conference*, Copyright 2016 by the author(s).

### 1.1. Early Methods

Deep Neural Networks have been rapidly maturing and being applied to many new and old machine learning applications and problem spaces. Many key ideas have been around for many years. Hebbian Learning (Hebb, 1949), the notion that the brain as a network of neurons, slowly learns based on some form of corrective feedback which adjusts neural firing parameters and synaptic weights. The perceptron (Rosenblatt, 1958), a probabilistic model for a Hebbian neuron using a simple activation function on inputs using a set of weights which could be used for classification or regression of various functions (equation 1). And finally, back-propagation (Werbos, 1974), or the iterative fitting of neuron weights using a loss function and chaining of loss gradients through a neural network (equation 2).

$$\mathcal{L}_{MSE} = \sum_{\forall i} (y_i - \hat{y_i})^2 = \sum_{\forall i} (act(x_i w_i + b_i) - \hat{y_i})^2 \tag{1}$$

$$w_i = w_i - \delta \frac{\partial \mathcal{L}_{\mathcal{MSE}}}{\partial w_i} \tag{2}$$

The key concepts of a network loss function such as mean squared error (MSE), and the use of the gradient back-propagation to tune neural network weights in perceptrons has been around for 40+ years, then why are things suddenly so different now?

### 1.2. Recent Advances

In more recent years we've seen the addition of convolutional layers (Le Cun et al., 1989) for visual invariance, the use of Dropout (Srivastava et al., 2014) as a powerful regularizer, the introduction of much faster gradient descent methods using RMSprop (Tieleman & Hinton, 2012) and Adam (Kingma & Ba, 2014), the use of Rectified Linear Unit (ReLU) activation functions (Nair & Hinton, 2010).

Other major recent advances included the introduction of the autoencoder architecture (Bengio, 2009) to provide unsupervised model learning or pre-training, much deeper stacked architectures of neurons and restricted boltzmann machines (Hinton et al., 2006), recurrent neural architectures such as the Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997), the introduction of powerful differentiable attention models (Mnih et al., 2014) (Jaderberg et al., 2015), and countless additional methods which arrive daily on arXiv and can't possibly all be summarized here.

Computational advances include the widely used CUDA (Nvidia, 2007) and CUDNN (Chetlur et al., 2014) GPU accelerator language tools, Neural Network architectures such Keras (Chollet, 2015), Theano (Bergstra et al., 2010), TensorFlow (Abadi et al., 2015), Caffe (Jia et al., 2014), and Torch (Collobert et al., 2002) which all provide collections of learning primitives, routines and automatic gradient computation for training with novel architectures and datasets. These have each played a massive role in realizing the scale of learning which is now required to partake in state of the art today.

### 1.3. Differences From Other ML Domains

Radio domain data presents several new challenges which differentiate it from many applications in the existing machine learning literature. Radio communications signals are quite well structured, and by their nature synthetically created by a man-made transmitter. Upon transmission however, they pass through many harsh channel effects fully imbued with the full chaotic and random goodness of nature. This creates a collection of variations present in received copies of an radio communications signals which create some unique difficulties.

Since communications systems are generally designed to operate near capacity limits, information bits are packed very densely, typically using a relatively simple set of basis functions or encoding, but not providing a lot of inherent redundancy or context since optimization for capacity is the norm. This is a stark contrast to the image domains in which spatial correlation and known objects make up much of the scene, as well as in speech recognition domains where relatively long recognizable formants occur only in known sequences to synthesize words and phrases, all of which have relatively robust a-priori characterized distributions and patterns which can assist significantly in error recovery.

Channel effects in wireless systems are a lengthy topic with a practically unending list of phenomena that occur and do interesting things to our signals in various propagation environments. Some of the effects that are present in virtually every real over the air system include:

- Random signal arrival times due to asynchronous traffic and protocol schedules as well as unknown propagation delays.
- Random symbol rate error due to free running sample clocks in detached radio systems.
- Random carrier phase and frequency error due to free running clocks, oscillators, and unknown phase-delay of various analog front-ends, reflectors, and other propagation medium.
- Non-Impulsive Delay Spread due to propagation multi-path interference, reflectors, and other effects.
- Doppler offsets resulting from motion of emitters, reflectors, and/or receivers
- Gaussian thermal noise, impulsive noise, co-channel and adjacent channel interference.

Some of these effects have equivalencies in other domains, some do not, and other have equivalencies - but with major differentiation. For instance, in voice recognition, random time of arrival, pitch(frequency) offsets, and word length dilation are all present, but they occur on much larger time scales and encode much less densely packed natural voice formants. On the other hand symbol meanings vary with time shifts often on the micro- or nano-second level rather than the milli-second level. Still, unknown time and dilation of a received signal can be considered a 1D Affine transform, not too different from attention models which can be used in imagery and voice recognition systems for such timing recovery.

Complex base-band representation commonly used in communications systems represents a fairly significant difference. We can treat this as with a multi-channel time-series in audio or finance, however there are certain properties such as random phase between channels where we lose basic polar representation properties when we treat them as independent channels. This makes tasks like phase recovery, frequency recovery, or Doppler/LO-tracking difficult on a basic representation level. We can introduce new primitive operations such as complex convolution as layers to help address this issue without needing to holistically address auto-differentiation in complex-valued neural networks, however this is still an open problem for including all complex operations.

Multi-path and equalization is another major issue; while present in voice recognition datasets it presents a more harsh problem in the communications domain due to the dense packing of information and the time scales over which multi-path components and symbols of information occur.

Since each of these common time varying random channel effects are present in most wireless systems, we do our best in this work to address and include these effects in datasets and when possible to make sure we are addressing a realistic description of each problem.

## 2. Building a Dataset

GNU Radio (Blossom, 2004) has many of the tools one needs to build a robust dataset in this field built in already. It includes a suite of modulators, encoders, and demodulators as well as a rich set of channel simulation modules (O'Shea, 2013) designed to apply simulated channel propagation models to these synthetically generated signals. At a high level, we simply string these logical modules together in GNU Radio to create our dataset.

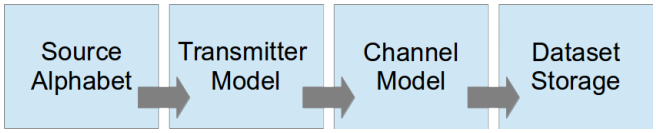


*Figure 1.* High Level Dataset Generation

### 2.1. Source Alphabet

We begin by selecting two data sources for all of our signals. For analog modulations we need a continuous signal such as voice, and so we use a publicly available copy of Serial Episode #1, which consists primarily of acoustic voice speech with some interludes and off times. For digital modulations, we use the entire Gutenberg works of Shakespeare in ASCII, with whitening randomizers applied to ensure equiprobable symbols and bits. All of our modems then use these two data sources with the corresponding digital or analog data source.

### 2.2. Signal Modulation

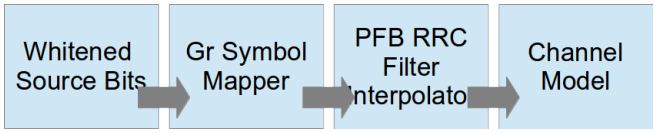


*Figure 2.* Modulation Process

We pick a normalized samples per symbol value to form a constant normalized symbol rate across all of our digital modulations. This is simple for single carrier modulations, but we must estimate a nominally similar value for bandwidth occupancy in multi-carrier systems such as OFDM and SC-FDMA. The goal is to make signals as similar as possible both in observed symbol rate and occupied bandwidth, these can then be varied in channel simulation and larger environmental simulations.

For PSK, QAM, and PAM, we implement transmitters using the gr-mapper (O'Shea, 2014) OOT module with a variety of constellations followed by an interpolating finite impulse response (FIR) root-raise cosine (RRC) filter to achieve the desired samples per symbol rate as shown in figure 2.

For GFSK, CPFSK, FM, AM and AMSSB GNU Radio hier blocks are used from mainline GNU Radio implementations. Before the final release of this dataset, in October, this list of modulation types is expected to grow.

### 2.3. Channel Simulation

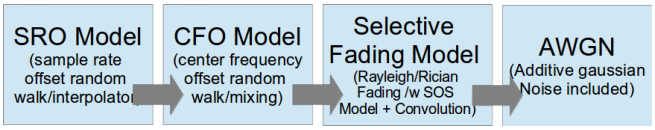


*Figure 3.* Channel Simulation Process

For channel simulation we use the GNU Radio Dynamic Channel Model hierarchical block. This includes a number of desired effects such as random processes for center frequency offset, sample rate offset, additive white Gaussian noise, multi-path, and fading.

These are each documented in more detail elsewhere in brief they consist of:

- Sample Rate Offset (SRO) Model: A fractional interpolator which steps along at a rate of $1 + \epsilon$ input samples per output sample, where epsilon is near zero, but follows a clipped random walk procesess to simulate sample clock offset/drift.
- Center Frequency Offset (CFO) Model: A digital oscillator and mixer which mixes the incoming signal off by a frequecy of  Hz, where this value is a clipped random walk process to simulate carrier frequency offset/drift.
- Selective Fading Model: As implemented in GNU Radio, implements the sum-of-sinusoids method with random phase noise for simulating Rician and Rayleigh fading processes on the incoming dataset with random time varying channel response taps.
- Noise Model (AWGN): The basic GNU Radio additive white Gaussian noise model which introduces thermal noise simulation at the receiver at a specific

noise power level corresponding to the desired signal to noise ratio.

### 2.4. Normalizing Data

Data normalization is an important step prior to machine learning use of the dataset. We would like to destroy any residual features which are simply an artifact of the simulation and would not reliably be present in a real world experiment. In this case, we ensure that each stored signal example is scaled to unit energy in each 128-sample data vector.

### 2.5. Packaging Data

We package the data such that it can be easily used in machine learning environments outside of the software radio software ecosystem. A simple method for doing this is simply to store the dataset as an N-dimensional vector using numpy and cPickle which is a popular format in the ML community for storing reasonably sized datasets.

We randomly sample time segments from the output stream of each simulation, and store them in an output vector. A commonly used tensor notation for Keras, Theano, and TensorFlow which we use here is that of a 4D real float32 vector, taking the form $N_{examples} \times N_{channels} \times Dim_1 \times Dim_2$. In this case we have $N_{examples}$ examples from the datasteam, each consisting of 128 complex floating point time samples. We treat this as $N_{channels} = 1$, a representation which commonly is used for RGBA values in imagery, $Dim_1 = 2$ holding our I and Q channels, and $Dim_2 = 128$ holding our time dimension.

Since radio domain operations are typically considered in complex baseband representation, which is not currently well suited for for many operations in ML toolboxes such as Theano and Tensorflow. We leverage the 2-wide I/Q $Dim_1$ to hold these in-phase and quadrature components in step with one another as a 2x128 vector. Since automatic differentiation environments for complex valued neural networks are not yet sufficiently mature, and a "complex convolutional layer" can obtain much of the benefit within this representation, we believe this representation is sufficient for the time being.

## 3. Signal Classification

Modulation classification is a classical machine learning/AI task in radio communications. In a dynamic spectrum access system, a spectrum access regulatory enforcement system, or any number of other diagnostic and monitoring scenarios, it forms an important task of labeling which emitters are present, available, and consuming spectrum in a given radio environment. Since this has relatively long been considered in literature using expert methods, it is an interesting jumping off point to compare non-expert methods driven by machine learning, and a task which we can relatively easily benchmark and make comparisons to prior work and approaches for these new methods.

### 3.1. Example Classifier and Performance

We provide an example classifier in Keras for the new dataset on github at `https://github.com/radioML/examples`. This represents the VT-CNN2 architecture optimized for the RML2016.04 dataset. It has not yet been tuned at all on 2016.10a. Performance is summarized below:

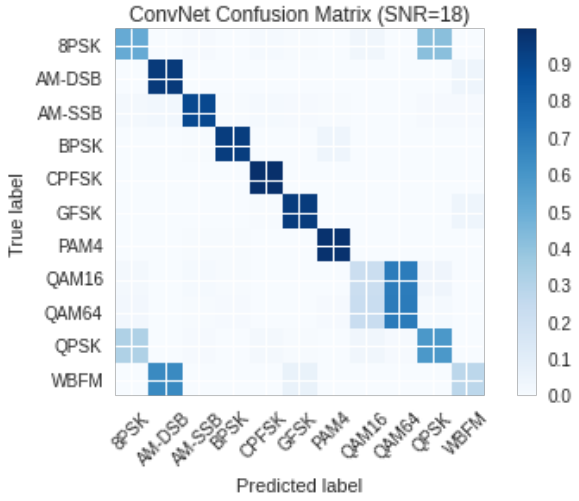


*Figure 4.* CNN2 High SNR Confusion on RML2016.10a Dataset

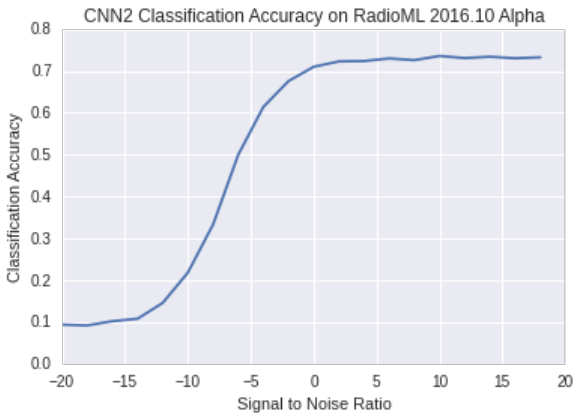


*Figure 5.* CNN2 High SNR Confusion on RML2016.10a Dataset

The functions in this example can be easily dropped into GNU Radio python blocks to form a streaming, online version of the tool. We expect this may take the form of an out of tree module in the future.

## 4. Other Radio ML Tasks

Modulation recognition is only one potential machine learning task in radio. Virtually any estimation, classification, regression, or transformation task can be treated as a machine learning problem and compared with prior methods for advantages and disadvantages. A number of the other applications we have considered thus far, but do not address in depth here are that of:

- Radio signal compression through learning sparse representations of the radio signal on a dataset. One hope is that by representing a signal sparsely enough, if it can be compressed to a representation on the order of the number of data bits in the signal, a signal decoder can be learned naively.
- Radio attention models, for automatically learning to synchronize and canonicalize a signal. To help remove channel state information reconstruction needs from compression tasks, as well as to allow discriminative classification or decoding tasks to learn to operate on a simpler canonical signal.
- Black box signal processing component regression training to replace existing modulation, mapping, randomization, coding or other discrete and well defined functions into a machine learning driven approximation of varying complexity.
- End-to-end communications system learning of new wireless channel encodings through channel autoencoders and data bit reconstruction cost across channel perturbations.
- Reinforcement learning for control of radio search, collaboration or tuning functions through active experimentation and manipulation of full waveforms.

These by no means represent a complete list of potential applications, but provides a few exciting starting points which have demonstrated some potential and viability for impact thus far outside of modulation recognition.

## 5. Parting Thoughts

The future of Machine Learning in the radio spectrum is really exciting. There are countless applications which can increase our ability to understand and act on the spectrum around us every day and improve how we represent, handle and automate our radio data tasks. GNU Radio and powerful Python based ML tools like Keras, Theano and TensorFlow provide a powerful combination of Software Radio, Signal Processing and Machine Learning tools which can be readily combined to accomplish many tasks at new state of the art levels of performance. We have shown here how GNU Radio can be used to produce high quality reference benchmark datasets for machine learning with known ground truth and harsh realistic channel assumptions.

As we continue to research the best methods for enabling many of these tasks, iteratively better, more complex, and more challenging datasets will be required to help compare, measure, and evaluate these tasks. Going forward we have already published a 2016.04 dataset, we will be publishing a 2016.10 dataset to coincide with GNU Radio Conference, and hoping to update and release new standard and easily named radio benchmark datasets periodically at least once or twice a year as needs and capabilities mature in GNU Radio and in the Machine Learning algorithm and application space. This will be done in an open way and all researchers in the field are strongly encouraged to contribute, critique, and discuss dataset needs and shortcomings to help shape this process.

## 6. Software and Dataset Availability

Datasets along with descriptions and a bit of boiler plate code for loading, unloading and performing some basic operations is available on the RadioML website. Datasets may be downloaded as pickle files from https://radioml.com/datasets/ and code to generate variations on them may be downloaded from https://github.com/radioML/dataset.

For ease of comparison we urge researchers to use the pregenerated pickle file to ensure consistent and comparable results. The generative GNU Radio model is provided so that special scenarios may be built, the generative process may be vetted and improved by third parties, and so that improvements and contributions from others may be included in future dataset releases.

## References

Abadi, Martın, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. *Software available from tensorflow. org*, 1, 2015.

Bengio, Yoshua. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific*

*Computing Conference (SciPy)*, June 2010. Oral Presentation.

Blossom, Eric. Gnu radio: tools for exploring the radio frequency spectrum. *Linux journal*, 2004(122):4, 2004.

Chetlur, Sharan, Woolley, Cliff, Vandermersch, Philippe, Cohen, Jonathan, Tran, John, Catanzaro, Bryan, and Shelhamer, Evan. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

Chollet, Franois. keras. https://github.com/fchollet/keras, 2015.

Collobert, Ronan, Bengio, Samy, and Mariéthoz, Johnny. Torch: a modular machine learning software library. Technical report, Idiap, 2002.

Hebb, Donald Olding. *The Organization of Behavior: A Neuropsychological Approach*. John Wiley & Sons, 1949.

Hinton, Geoffrey E, Osindero, Simon, and Teh, Yee-Whye. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jaderberg, Max, Simonyan, Karen, Zisserman, Andrew, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pp. 2017–2025, 2015.

Jia, Yangqing, Shelhamer, Evan, Donahue, Jeff, Karayev, Sergey, Long, Jonathan, Girshick, Ross, Guadarrama, Sergio, and Darrell, Trevor. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

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

Le Cun, Yann, Jackel, LD, Boser, B, Denker, JS, Graf, HP, Guyon, I, Henderson, D, Howard, RE, and Hubbard, W. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

Mnih, Volodymyr, Heess, Nicolas, Graves, Alex, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pp. 2204–2212, 2014.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

Nvidia, CUDA. Compute unified device architecture programming guide. 2007.

O'Shea, Tim. Gnu radio channel simulation. In *GNU Radio Conference 2013*, 2013.

O'Shea, Tim. gr-mapper. https://github.com/gr-vt/gr-mapper, 2014.

Rosenblatt, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

Werbos, Paul. Beyond regression: New tools for prediction and analysis in the behavioral sciences. 1974.