Chapter X
Sub-Signal Detection from Noisy Complex Signals Using Deep Learning and Mathematical Morphology

Jie Wei¹, Hamilton Clouse², Ashley Diehl²
¹Dept. of Computer Science, The City College of New York, NY, jwei@ccny.cuny.edu
²Air Force Research Laboratory, Wright-Patterson Air Force Base, Fairborn, OH

Contents
X.1 INTRODUCTION 1
X.2 LSTM-RNN AND MATHEMATICAL MORPHOLOGY BASED ALGORITHM TO DETECT
SUB-SIGNALS FROM NOISY COMPLEX SIGNALS 4
1. Data preparation and pre-processing 5
2. LSTM-RNN local sub-signal learning 9
3. Mathematical morphological global sub-signal detection 11
X.3. EXPERIMENTAL RESULTS 14
X.4. CONCLUSION 20
References

X.1 INTRODUCTION
In recent years, Artificial Intelligence, especially machine learning, and deep learning in particular, has revolutionized intelligent vehicles considerably. The development of Convolutional Neural Net (CNN) [1], Deep Belief Net (DBN) [2], Stacked Auto-Encoder (SAE) [3], and Long Short-Term Memory (LSTM) based Recurrent Neural Net (RNN) [4] over the last several decades have enabled the pursuit of semi-autonomous/autonomous vehicles and networks of intelligent vehicles. The availability of sufficient and balanced training data is a crucial factor behind every success story of a deep learning approach. One such approach was the deep reinforcement learning based AlphaGo [6]: AlphaGo repeatedly beat reigning human Go champions with ease. The researchers accomplished this feat by using millions of existing Go games to train and re-train itself by participating in thousands of self-playing games per day to improve its policy at each step; this is more games in a day than a human professional player can play for their whole career. Another astounding deep learning approach was a CNN-based skin cancer classifier developed by researchers in Stanford University: The network used nearly 140,000 skin lesion images to achieve expert level performance when identifying cancerous lesions [7]. To exploit the immense power of deep learning, great efforts must be made to produce the large, balanced training sets. As it is expensive to collect and label large amounts of representative data, we must also harness transfer learning, i.e., use a small, labeled dataset to re-train and fine-tune a subset of a deep network architecture that has already been trained using relevant big data [8].

The increase in intelligent vehicles on the road increases the number of vehicle-to-vehicle and vehicle-to-infrastructure communications. Protecting these communications is becoming increasingly important. This is true whether you have intelligent civilian vehicles on the road or intelligent military vehicles across the ground and sky. In these intelligent vehicles, to effectively protect the drivers, passengers, and facilities and make timely response to potential signals from surrounding environments, we identify and characterize
possible signals, often-times embedded in the signals of complex time series contaminated by severe background noises due to intentional camouflage and information hiding schemes, sensors’ and/or aircrafts’ unstable and random movements and vibrations, and other unpredictable environment and climate sources. In Fig. x.1, three synthetic signals with known sub-signals are depicted. The rampant noises present in the signals makes a great challenge to reliably detect the presence of sub-signals from within the severe noises. In Fig. x.1a, the step-like sub-signal waveform, depicted in the bottom row, contains signal magnitudes around 0.001 and is contaminated by noise with magnitudes larger than 0.0020; the resulting magnitudes of the sensed signals, depicted in the middle row, are of magnitudes around 0.0023. In Fig. x.1c, the sub-signal waveform has extremely high frequency; its separation from surrounding noise is also hard to detect. The sub-signals in both cases shown in columns (a) and (c) are difficult for human operators to identify and characterize; this is especially true when attempting to depict the accurate start and end of the presence of every sub-signal due to the strong noises. The sinusoidal sub-signals loaded in signals shown in Fig. x.1b is probably the easiest to identify by human operators. Using methods such as conventional statistical signal processing or Fourier analysis can often be used to detect these types of sub-signals [9]. Since these types of “easy” cases are infrequent, the ad-hoc signal processing and Fourier analysis based approaches that are tailored to detect these easy cases will fail in the presence of hard cases, as seen in our experimental studies.

Despite great progresses made in time series analysis [10] and deep learning [11], due to the immense noises, widely varied waveforms of sub-signals, and lack of straightforward deep learning approach to process complex numbers directly, the efficient and reliable method to detect these complex sub-signals remains elusive. In response to this challenge, researcher engineers and scientists from the Air Force Research Laboratory and the City College of New York, have taken up this problem and worked closely together to apply our prior knowledge and experience in signal processing [12], machine learning and deep learning [13, 14] to detect the sub-signals from among rampant noises. Given that the signals are 1-D
complex sequences, it is appealing to apply various mature time series processing and analysis schemes [10, 15] such as Autoregressive moving average (ARMA), Autoregressive Integrated Moving Average (ARIMA), spectral analysis, hidden Markov Model (HMM) and dynamic programming [16], which the authors have developed a slew of different methods with varying degrees of success in prior researches [17-20]. However, our initial endeavors to harness these otherwise powerful time series processing and analysis schemes to identify and characterize these sub-signals were unable to combat the challenges posed by this problem, due mostly to the heavy noises innate to the signals, the widely different nature of the waveforms of the sub-signals and irregular and unpredictable presence and absence of sub-signals, and the sophisticated inter-plays of all these troubling factors.

One major drawback of signal processing methodologies is the reliance on underlying signal and noise models to be addressed, the sub-signals that may arise in practice are so varied that it is impossible and methodologically incorrect if we made any assumptions on the waveforms, the nature of the system, and random noise. The only viable approach is using machine learning techniques [8, 21]; these approaches make as few assumptions as possible regarding the problem in hand and let the data speak for itself. In our previous work, machine learning methods such as shallow neural networks, logistic regression, and random forests have been exploited to achieve promising performances in image denoising [22], civilian vehicle engine detection [14], breath signal detection [23], 1D breath signal analysis [24], and MRI super-resolution [25, 26]. However, there is no conventional machine learning methods that can effectively take advantage of the locally contextual dependencies in the sub-signals loaded signals, i.e., sub-signals are present as connected clusters, and widely different sub-signals waveforms as illustrated in Fig. x.1. Unlike ordinary neural networks where the inputs and outputs are considered independent, RNN takes the outputs in prior step(s) as part of the inputs, thus explicitly considering the contextual dependencies in the temporal data. Although theoretically a better network, the recurrent cycle from prior outputs of RNN only has the short-term memory. Only the outputs of the preceding one or several steps can be practically remembered; long-term dependencies, which is far from unusual in time series and language models, cannot be facilitated by this network architecture. The representational and predicting power of RNNs was significantly enhanced with the introduction of LSTMs—a miniature neural network on its own—that can selectively remember and forget previous memories in the local contexts; these keep relatively long-term dependencies with reasonably small additional units [4]. LSTM-based RNNs have been one of most widely used deep learning methods and have found a great array of utilities in many applications such as natural language processing (NLP) [27] and time series analysis [28, 29] with desirable performance.

After exploring several research avenues, from statistical signal processing to shallow machine learning with subpar results, we eventually set out to apply LSTM-based RNN (LSTM-RNN) in our sub-signal detection work. After careful problem formulation and data preprocessing, the original noise-loaded signals are turned into a large volume—from hundreds of signals to the order of ~100,000—of balanced sub-signals training instances, the LSTM-RNN N is trained to enhance and predict the presence of sub-signal in each location. For each incoming signal segment s, after going through the same data pre-processing step as done in N training phase, s is enhanced by use of N. The resulting signal enhanced the sub-signals signals while suppressing noises in the original data thus significantly increased the signal to noise ratios. A post-processing procedure based on mathematical morphological (MM) operators such as watershed, run-length encoding and size filtering, can be applied to segment s to arrive at the global shape/signature of the sub-signals in s. These MM operators can effectively encode the global properties of sub-signals in the data segment s. As observed from the sub-signals simulation procedure we were provided, this new combination of LSTM-RNN’s immense learning power to predict local presence of sub-signals and MM’s prowess to enforce global signature has given rise to encouraging results of primary interest to ensure the security of civilian or military intelligent vehicles.

In the next section, the technical details of the sub-signals detection algorithm are described. The algorithm components described include pre-processing, LSTM-RNN training, and MM-based global post-processing. Experimental results for signals generated by sub-signals simulation procedures are reported
and analyzed in Sec. x.3. We conclude this paper in Sec. x.4 with more remarks on possible new directions to be taken in the near future on this sub-signal detection work for intelligent vehicles.

**X.2 LSTM-RNN AND MATHEMATICAL MORPHOLOGY BASED ALGORITHM TO DETECT SUB-SIGNALS FROM NOISY COMPLEX SIGNALS**

In this section, the technical details of our LSTM-RNN and MM based algorithm to detect sub-signals from noisy complex signals are provided. The sub-signals signals used in this study are generated using simulation functions that serve as faithful proxies of practical sub-signals signals based on extensive prior exploratory data analysis over sub-signals signals. To represent the real-world situations as closely as possible, the simulation procedure in MATLAB was made available to this research team, where the specific waveforms and durations, Gaussian smoothing factors, and noise levels that may arise in practical signals of interest are generated by random sampling [30]. Each simulated data segment \( s \) generated by these simulation functions is a sequence of 1,600 complex numbers, the ground truths of the sub-signals in each \( s \), namely, the waveform and precise duration of each sub-signal are also available for ease of algorithmic developments.

Three such segments are illustrated in Fig. x.1, where top row are the simulated signals, and the next two rows provide the known ground truths about the waveforms – middle row – and presence of sub-signals, bottom row. The objective of our algorithmic development is to take signals \( s \) as inputs to generate outputs corresponding to the presences \( t(s) \) of sub-signals, where \( s \) is a 1600-dimension vector, and \( t(s) \) is the corresponding 1600-dimension Boolean (indicator) vector indicating the presence of sub-signal at each location in signal \( s \). The overall logic flow chart of the proposed algorithm is diagrammatically depicted in Fig. x.2. Technical details of each step in both training (colored in red or green) and testing (in green or blue color) phases are expanded in detail in the sequel.

![Logic flow chart of the proposed algorithm](image_url)

**Fig. x.2** Logic flow chart of the proposed algorithm. Training phase (red data flow): training data go through data preparation and preprocessing and LSTM-RNN local sub-signal step to deliver the LSTM-RNN local predictor. In testing phase (blue data flow), the testing data first go through data preparation and preprocessing procedure and then the signal is enhanced by the trained LSTM-RNN local predictor; the final sub-signal detection results are declared by the MM-based global post-processing step. Color designations: Data and procedure associated with training phase only are colored in red; while those corresponding to testing phase only are in blue color. Procedures involved in both phases are colored in green.

**X.2.1 DATA PREPARATION AND PRE-PROCESSING**
As briefly introduced in Sec. x.1, to effectively harness the great power of deep learning that is extremely data hungry, it is crucial to have a large and balanced data as the training data. Our data preparation and preprocessing step is developed to generate large and balanced training data that can effectively reflect the targeted sub-signals for effective following processing.

By large data, for every class to learn, sufficient number of instances should be available. For instance, in LeCun et. al.’s seminal effort of digit recognition using CNN [31], to learn the ten digits 60,000 28x28 training digits were provided as the now-standard MNIST dataset. As reported by Stanford’s skin cancer detection using CNN [7], their excellent skin cancer performance, better than or at least in par with human dermatologists, are only possible after they have adequate training skin lesion images in the order of a hundred thousand (~140,000 to be precise). In our sub-signal detection work, it is impossible to learn the 1600-dimension sub-signal indicator $t(s)$ directly from 1600-dimension signal $s$: the extremely high dimension (1600) of the objective to-be-learned targets will in theory demand prohibitively more training data, and more importantly, the sub-signals, as shown in bottom row of Fig. x.1, can present essentially everywhere in $s$ with totally random duration, which is determined by drawing a random variable in the simulation procedure, except for a few weak constraints about the behavioral property of the sub-signals to be described elsewhere in Sec. x.2.3—which will be exploited in the global post-processing phase—there is no global regularities or interplays among the presence or absence of sub-signals and different waveforms that can be learned from the training data, however big they may be. Therefore, it is an ill-posed problem to learn the entire 1600-dimension $t(s)$ directly from $s$ as the space spanned by possible $t(s)$ could be in an order exponential to 1600 as the presence and absence of sub-signal is randomized.

A problem re-formulation is thus needed to pose the sub-signal detection problem as a viable deep learning problem: if there is little global regularity to be learned for the entire $t(s)$ from the global data segment $s$, we can then single out the individual Boolean indicator $t_i$ from $t(s)$, $i$ ranging from 1 to 1600, and try to learn the 1D Boolean value from $s$, again due to the random nature of the signal $s$ in real sub-signals signals and reflected in the available simulation functions, it is unlikely $t_i$ is determined by the entire $s$. Conversely, the Markovian neighborhood can be safely assumed [17], i.e., $t_i$ can be viewed as a function of its local neighborhood $N$ in signal $s$ only: $s[i-k_1 : i+k_2]$, that is, the vector of size $k_1+k_2$ centered at location $i$, the position of $t_i$. Because the signal is non-causal, the reversal of $s$ remains a legitimate signal, in this work we thus have no reason to have different $k_1$ and $k_2$ values, hereafter they are denoted by one value $k$. After intensive empirical studies, we found that setting $k$ at 20 gives rise to good overall performances for datasets generated by the simulation function. In consequence, the new learning problem is to learn a Boolean indicator function $f$ for scalar $t$ from $2k$ vector $v = s[i−k, i+k]$, as dictated by the equation below.

$$t = f(v) \quad (x.1)$$

From this problem formulation, each 1600-dimension signal $s$ can yield up to 1600-2k (by trimming the start and ending positions in $s$), given that the $k$ is set at 20, 1560 training instances can be formulated to learn $f$ for Eq. (x.1). Since there are only two classes, i.e., presence of absence of sub-signal, following the example of other deep learning applications, e.g., [1, 31], one hundred typical $s$ will give rise to sufficient training data for deep learning purposes, with an order of ~100,000 available training instances. Thus, the large data size requirement for deep learning can be facilitated with this problem formulation.

The other requirement for effective deep learning is the balance of training data. In our problem as dictated by Eq. (x.1), the positive and negative responses of $t$, the actual presence and absence of individual sub-signal, should be roughly balanced: if there are too many positive (negative) instances, the learned deep net will unfairly favor positive (negative) instances thus reducing the quality of the net considerably. In our signals, there are many cases that the sub-signals are only present in a very small portion of the signal $s$, as seen in Fig. x.1c, only about 10% of $s$ are positive instances for $t$. Even worse, there are many more signals where there is no sub-signal at all, which is the norm in practical scenarios. To avoid this imbalance
problem, in preparing the training data, instead of using all s to obtain t’s and v’s, cf. Eq. (x.1), the following two choice rules are applied to enforce the rough balance of positive and negative training instances:

Choice rules:

1. s is discarded if there is no sub-signal
2. for every positive sub-signal chunk s_t, of length L, only keep possible 0.6*L prior and post non-sub-signal locations in s

Rule 1 ensures that those signals with no sub-signal presence should not be employed for training at all. Rule 2 makes certain that the number of positive and negative t’s are roughly of the same number: we intentionally keep 60% prior and post non-sub-signal positions because the following two factors that will reduce the actual number of negative instances: 1. If a sub-signal chunk is close to the starting or end of signal s, there will be fewer negative instances; 2. the next skipping in the leading and ending locations in forming the local neighborhood v for each t will reduce the number of actual negative instances. The potential imbalance trouble in training data is thus resolved. Because of these choice rules which reduced the number of instances, to ensure sufficient number of training instances, two to three hundred 1600-dimension signals are generally enough to produce ~100,000 balanced training instances. In our tests, ~250 signals are harnessed to train our LSTM-RNN to predict local sub-signal.

Mathematically, the procedure to find t from v as dictated by Eq. (x.1) is a typical inverse problem [32], whereof special care should be taken to avoid troubles of this generally ill-posed problem. In view of the severe and rampant noises present in the sub-signal signals, to effectively enhance sub-signals while suppressing noises, the regularization technique, which was developed in applied mathematics to help solve inverse problems, has been widely used in image processing [33] and machine learning [34] with promising results. The key to the regularization technique is the introduction of additional information of the desirable properties of the objectives. For instance, in the celebrated snakes or active contour models [34], the internal energy is added in the optimization process to enforce the continuity and smoothness, the gradients (first partial derivatives) and curvature (second partial derivatives) of the desirable contour to be sought after. In our detection problem, along the line of the internal energy addition in active contours, the desirable and/or known property of sub-signals should also be encoded as a regularization term to achieve better detection results. The 2k-dimension data segment v in Eq. (x.1) is the sum of the contribution of sub-signal 2k-dimension st and system and/or 2k-dimension random noises n:

\[ v = st + n \]  

(x.2)

The additional regularizing terms should emphasize the property of sub-signals that are significantly different from noises. After checking all cases of the available simulation functions Gen00 to 05, as illustrated in Fig. x.1, we found the common property shared by all sub-signal signals: although of varying frequencies, the sub-signals st are still smoother than those noises n within the relatively short duration 2k, notice that k is ~20. To encode the smoothness contributed by st, the Fourier transform fits in perfectly: now that st is smooth, its corresponding Fourier transform should be mostly dominated by its low frequency components; conversely, noise vector n’s Fourier transform should contribute far more high-frequency components, that is, if we take the Fourier transform on both sides of Eq. (x.2), the linearity of Fourier transform will result in the equality below:

\[ F(v) = F(st) + F(n) \]  

(x.3)

Since the low frequency components of F(v) mostly come from F(st), if only the top m AC components are kept—the signals used in this work are zero-mean, hence the DC component F(v)[0] is always 0 and thus ignored—the following approximation is obtained:

\[ ACm(v) \approx ACm(st) \]  

(x.4)
where AC_m(x) = F(x)[1:m], the first m AC components of x. Hence the smoothness regularization term due to sub-signal st can be safely represented by the leading m AC components of the Fourier transform of v. The resultant regularized term r(p) at a position p is below

\[ r(p) = [v(p), ACm(v(p))] \]  

which is a vector of 2k+m dimension. From our experiments, optimal detection performance occurs when m takes value 3, hence for each position p, as k=20, we need 43-D vector in order to train a learner for sub-signal t. Because in our following LSTM-RNN training process, the L2 norm is used, Parseval’s theorem [10] ensured that the sum of squares of AC_m(v(p)) is (approximately) equal to that of st. r(p) is therefore a viable regularized representation at location p to be optimized and learned with desirable smoothness of sub-signals effectively encoded. Although in theory different weights should be assigned to different AC coefficients given their different nature. From our tests, likely due to the widely different sub-signals, putting these weights higher or lower than 1 cannot yield better performances over all six different simulation signals. In this work we simply assign all three weights to 1.

One more important issue needs to be addressed before engaging LSTM-RNN learning: the type of number system. As mentioned in Sec. x.1, all signals in the sub-signal identification and characterization tasks are complex, thus the 43-D vector r(p) for each position p is complex as well. However, so far there is no readily available support for the use of complex numbers available in TensorFlow, likely due to the lackluster performances. Despite the theoretical advantage the complex representations can offer, as reviewed in Bengio et. al.’s recent unpublished manuscript [29] (submitted to NIPS in May 2017), most recent deep learning work using complex numbers directly can only handle toy tasks, with very few exceptions [35]. In the work reported here, two different approaches are adopted to convert the 43-D complex vector r(p) to real numbers that can be handled by current TensorFlow backend that we have been using in this work. 1) The first one is to use the magnitude of a complex representation:

\[ \text{ra}(p) = ||[v(p), ACm(v(p))]|| \]  

where | c | is the magnitude, \( \sqrt{c_{\text{real}}^2 + c_{\text{imag}}^2} \), of a complex number \( c = c_{\text{real}} + c_{\text{imag}} i \). Thus the real value vector ra(p) consists of 43 positive real numbers that are the corresponding magnitudes of the complex numbers. 2) The second one is to use the absolute values of both the real and the imaginary components of a complex variable:

\[ \text{rb}(p) = \text{abs}([\text{real}(v(p)), \text{imag}(v(p)), \text{real}(ACm(v(p))), \text{imag}(ACm(v(p)))]) \]  

where real(c) and imag(c) are the real (c_{\text{real}}) and imaginary (c_{\text{imag}}) part of a complex variable \( c = c_{\text{real}} + c_{\text{imag}} i \). The abs(•) operator takes the absolution value of the real numbers. The vector rb(p) is composed by 86 positive real numbers for the corresponding 43-D complex vector. According to the meaning of complex representations, each component in ra(p) indicates the radius of the complex number, thus all complex numbers of the same radius are mapped to the same value; whereas in rb(p) the absolute values of the real and imaginary part are used in the representation: only four complex numbers can be mapped to the same two values in this case—by folding four quadrants to the first quadrant, therefore, the many-to-one transform employed in representation ra(p) is not as drastic as that in rb(p), thus the former can potentially load more information. As to be reported in Sec. x.3, without the abs operator the detection accuracy is unacceptably worse probably due to the numerous periodic sudden phase changes in the complex representations that are irrelevant to the presence or absence of sub-signals, which could indicate some complications in direct complex number usage. Sub-signal detection performances for both representations based on Eqs. (6, 7) will be reported in Sec. x.3.

**X.2.2. LSTM-RNN LOCAL SUB-SIGNAL LEARNING**

The careful data preparation and pre-processing step as described in the preceding section generated a big, balanced and regularized dataset, namely the ra’s and rb’s, which are 43- or 86-D real vectors,
respectively, that can be processed by all deep learning packages. In view of the new regularized data representation, in place of Eq. (x.1), for each position p, the sub-signal t(p) is a function g() of r(p), which is to be learned by a deep learning process:

\[ t(p) = g(r(p)) \]  \hspace{1cm} (x.8)

where r is either \( r_a \) or \( r_b \). The memory unit (forget/remember gate) introduced in LSTM is the only means to fully take advantage of the potential long-term temporal dependencies among the sequential data in r(p), a 43- or 86-dimension vector. Because of this, LSTM-RNN, the LSTM unit based RNN, has been widely used in temporal/sequential data processing. In this work, after trying out other deep learning schemes, LSTM-RNN is found to yield optimal performances consistently, which indicates that there indeed exist long-term dependencies in the sub-signal signals which can be only captured by the LSTM memory units. This specific net is thus employed in this work as the workhorse.

The next two choices to be made on the nature of LSTM-RNN are number of layers and regression or classification for the target t as dictated by Eq. (x.8). Unlike visual object classification tasks taken up by CNN or RNN [1, 7], where a lot layers are needed to find edges, contours and local semantically meaningful shapes that contribute to the formulation of a visual object, in our sub-signal signals within a relatively short span of 43 or 86 numbers, the hidden sub-signals do not have sophisticated composite curves or shapes to be discriminated, therefore the number of layers, generally accounting for contextual structures of different semantics, needed for this sub-signal detection task is not too large. From our empirical studies, we found one layer of LSTM is adequate in this RNN, additional layer(s) of LSTM cannot yield better performances.

The other important choice is 1) direct classification: to set the LSTM-RNN to directly learn class label, presence or absence of sub-signal, for t(p) from r(p), thus yielding an essential full-fledged classification machine; or 2) regression as a data enhancement procedure: to use the LSTM-RNN as a regression machine to enhance the hidden sub-signal signals while suppressing the rampant noises that will be relayed to the next post-processing step for final sub-signal classification. Due to the significantly different sub-signal waveforms—ranging from step curves to sinusoidal of different frequencies and Gaussian shapes—and even more different noise levels—from Gaussian, uniform to flicker noises of entirely different seriousness, as evidenced by our intensive empirical studies, the direct classification scheme can only generate subpar performances for our data.

In the second option the LSTM-RNN plays a role as a regression step to enhance the targeted sub-signal signals while suppressing the severe noises thus achieving data enhancement, this seemingly modest approach of the LSTM-RNN application has been employed recently by several groups for effective signal enhancement purposes recently with promising performances [36, 37]. From our empirical studies, we observed that while it is extremely difficult to train a LSTM-RNN to come up with acceptable sub-signal classification labels for different types of sub-signal signals which is highly correlated to the local sub-signal waveforms and noise natures, the local long-term dependencies captured by the LSTM units can enhance the sub-signals and suppress noises by varying orders of magnitude: in some cases the LSTM-RNN can almost boost sub-signals to 1 and suppress background noise to 0 thus behaving like a great classifier; whereas in many cases the sub-signals are elevated to modestly larger values than those of background noise, e.g., \(-0.035\) for sub-signals and \(-0.025\) for background signals. This LSTM-RNN enhanced data will provide a solid foundation for our next post-processing step to finally classify sub-signals by taking account of the global behavioral properties of sub-signals in the signal segments, which is impossible to be effectively learned by LSTM-RNN given that this global properties are related to dependencies of distance in terms of up to several hundred even a thousand data points, well beyond the dependencies a LSTM-RNN can effectively handle and thus making this line of work impractical. In addition, we believe for this type of global analysis classic signal processing, especially mathematical morphological [38], fits in perfectly and thus should be exploited.
The layout of the training phase of the LSTM-RNN regression model for data enhancement, the actual main graph generated by TensorBoard coupled with the TensorFlow package, is shown in Fig. x.3, the input 43- or 86-D data are fed to the LSTM layer; the outputs of this layer are fed to a fully connected logistic neural net (tanh activation function is used in place of the default sigmoid with faster convergence time and similar results). The target data are compared with the output of the net to update the learned parameters. This rigorously learned model will be employed in the testing and detection phase to enhance data quality thus laying a solid foundation for the next global sub-signal detection step, as empirically proved by our experimental studies presented in Sec. x.3. The two different conversion methods for complex numbers have almost the same architecture except the dimensionality of the input data. This trained model is saved as the sub-signal local predictor depicted in Fig. x.2, to be used in the detection phase. The training phase, the red colored flow in Fig. x.2, is now complete.

Fig. x.3. Architecture of the trained LSTM-RNN model, the actual main graph generated by TensorBoard coupled with the TensorFlow package. The SGD (stochastic gradient descent) is used as the optimization approach.

X.2.3. MATHEMATICAL MORPHOLOGICAL GLOBAL SUB-SIGNAL DETECTION

The sub-signal detection task is performed over the testing data, the data flow designated in blue color in Fig. x.2 by using the LSTM-RNN predictor for data enhancement. The incoming 1600-D data first go through the pre-processing step—without applying the two choice rules which are called upon for the sole purpose of balanced training data and should thus be skipped in the detection phase—to generate the 43- or 86-dimension representation vector \( r(p) \)’s, cf. Eq. (x.5). These \( r(p) \)’s are next fed to the trained LSTM-RNN local predictor for data enhancement, which can effectively suppress noise while boosting sub-signal signals. The signals enhanced by the LSTM-RNN model for the three noisy signals based on \( r_b \) shown in Fig. x.1 are displayed in Fig. x.4, the three sub-signals revealed themselves more clearly from the background noise more easily than the original ones.
Fig. x.4. The three signals shown in top row of Fig. 1 are enhanced by the learned LSTM-RNN local predictor using \( r_b \), the sub-signals whose ground truths given in bottom row of Fig. 1 in the middle, left, and right parts, respectively, are far more outstanding than the original ones after this process. The values are normalized for each 1600-D segment.

Because in the training process the presence and absence of local sub-signal takes value 1 and 0, respectively, and the addition of three leading AC coefficients in the trained representational vector \( r(p) \)’s, as evidenced in the three enhanced signals, the values corresponding to sub-signals tend to have higher values than those associated with background noises. The detection of sub-signals is thus the separation of foreground from the background from the 1600-D data segment using certain global properties of the foreground, mathematical morphological (MM) operators fit perfectly in this task, which the authors have utilized expansively in lots of image processing [22, 39, 40], computer vision [40, 41], and medical imaging applications [9, 13, 26] with promising performances to segment out moving objects, human beings, brain vessels, and lungs. To effectively apply MM approaches, the global nature of the sub-signals should be first laid out. As briefly discussed in Sec. x.2.1, although the sub-signals did not exhibit too much global regularities given their randomized nature, they nonetheless have some weak but useful statistical properties for MM operators to exploit effectively for detection purposes:

1. **Size property:** to convey meaningful information, the length of each sub-signal is of considerable size. Indeed, after running the simulation functions multiple times, on average more than 99.9% sub-signals are of length larger than 60, hence any cluster of foreground whose length is less than 60 can be dismissed as background.

2. **Sub-signal sparseness property:** the sub-signals did not present themselves as dense pulses or sinusoidal curves. If the gaps between two possible foregrounds should be merged as one because of the sparse nature of sub-signals. In addition, the sparseness property also demands that, within a signal segment, if there are many candidate sub-signal detection results, the one with the smallest number of sub-signals should be preferred. This rule of thumb can effectively combat the false positives caused by severe noises present in sub-signal signals.

The foregoing two statistical properties are justified by the available sub-signal simulation functions and researchers with knowledge of real sub-signal signals from applications. They serve as guidelines to the
ensuing MM based sub-signal detection processing, a 1-D watershed based segmentation based on run-length encoding.

Given the different foreground sub-signal waveforms and noise levels, it is impossible to find a global threshold to classify the enhanced signals. Instead, careful considerations should be given to the nature of each segment individually. The two foregoing properties of authentic sub-signals point to a powerful MM segmentation methodology, the watershed transform [42]. Simply put, the idea is to tentatively flood the landscape from minimum values and find a set of barriers when different sources of flooding meet, the segmentation from this set of barriers that optimizes the objective is the resultant segmentation. To speed up the segmentation, which is crucial for practical use, a simplified procedure is applied due to the nature of sparseness instead of minimizing the weighted sum of variances in Ostu’s method, we used

\[ tp(\delta) = \text{senh} \geq \delta \]  

(x.9)

The possible classification result \( t_{p(\delta)} \) due to \( \delta \) is a 1600-D binary vector. The ideal classification result \( t \) should be the one best satisfying the two above-mentioned sub-signal properties. To determine the optimality of a candidate \( t_{p(\delta)} \), the tentative foreground and background sequences declared by \( \delta \) in \( t_{p(\delta)} \) should be checked: 1) the size of declared sub-signals, that is, the continuous foreground clusters, should have considerable size (due to the size property); 2) if two foreground clusters are too close, or put another way, a background sequence in-between two foreground clusters is too short, then the two foreground clusters should be merged as one (due to sparseness property). The run-length encoding (RLE), a powerful binary encoding algorithm widely used in signal processing and image compression, is employed to facilitate the foregoing optimality checking: after performing RLE procedure, the nature (0 or 1) symbol of each run and the corresponding number count of positions for each are available: if there are \( n \) runs in \( t_{p(\delta)} \), symbol\([i] \), \( i=0..n-1 \) stores the \( i \)th run is 0 or 1, while count\([i] \) stores the length of the \( i \)th run. The following steps are conducted in order for optimality checking:

**Procedure Optimality checking of \( t_{p(\delta)} \):**

1. Apply binary MM opening and closing operators to cleanse isolated foreground and background chunks (to result in shorter RLE codes);
2. Perform RLE to obtain run length array symbol and count.
3. if symbol\([i]=0 \) and count\([i] < \text{min(count}[i-1], \text{count}[i+1]) \), merge three runs No. \( i-1, i, \) and \( i+1 \) to a single 1 run;
4. if symbol\([i]=1 \) and count\([i]<60 \), merge three runs No. \( i-1, i, \) and \( i+1 \) to a single 0 run.
5. Conduct another RLE and return the new symbol and count arrays as outputs.

The \( t_{p(\delta)} \) that yield the smallest number of foreground cluster(s), evaluated by use of the symbol and count arrays returned by the preceding procedure, is chosen to be the best classification results \( t \), this rule is applied due to the second part of the sparseness property. If there is more than one \( \delta \) yielding the smallest number of foreground clusters, the one corresponding to the largest \( \delta \) is opted for, which is based on the heuristic that results due to larger \( \delta \) is less likely to be disturbed by severe noises. The sub-signals for the three signals detected by this MM-based post-processing procedure using \( r_b \) are presented in Fig. x.5, the accuracy, precision, recall, and \( f_l \) scores of these three detection results are (0.99, 0.99, 1.0, 0.99), (0.99, 0.92, 1.0, 0.96), (1.0, 1.0, 1.0, 1.0).

---

2 This is more like the exhaustive threshold searching procedure in optimal threshold discovery, the difference is, instead of minimizing the weighted sum of variances in Ostu’s method, we used size property and foreground sparseness as the objective function.
X.3. EXPERIMENTAL RESULTS

To inspect the performances of the proposed algorithm developed in the preceding section and summarized in the flow chart of Fig. x.2, the simulation function available to us are employed to generate the training and test cases. It is worth noting that these simulation functions are entirely stochastic, i.e., the specific waveforms, start time and duration, and noise type and levels are all randomly drawn; the only parameter that can be deterministically controlled is the number of instances to be generated. In consequence, every different call of these functions will yield entirely different sub-signal data contaminated by different noises. We first call it to generate sub-signal signals as the training data. As described in Sec. x.2.1, 200 to 300 data files will be sufficient for the purpose of producing a big and balanced data representation of’s for deep learning. We thus generate 250 sub-signal signals to serve as the training set. After going through the choice rules and transforms dictated by Eqs. (3-5), which is the red data flow in Fig. x.2, ~140,000 43 complex valued instances are available for the training phase delineated in Sec. x.2.2, using either the complex magnitudes or the absolute values of the real and imaginary components, dictated by Eq. (x.6) and Eq. (x.7), respectively. In this work, python 3.6.1 from Anaconda 4.4.0 platform, TensorFlow 1.1.0 with GPU support and TFlearn 0.3.2 API are used for coding development. The two trained LSTM-RNN data enhancement model for both complex value encoding schemes are saved for the testing phase. The experiments reported in this section are conducted over a Dell Precision T7610 desktop computer with Intel Xeon CPU E5-2630 v2 with 2.60GHz and 32.0 GB on Windows 10 OS. The training of the LSTM-RNN with 150 epochs for the two proposed algorithm costs 95 min and 219 min for methods, respectively. The average detection time for one 1600-D test data item is around 0.06 sec and 0.15 sec, respectively. If code optimization techniques such as Numba and Cython and more powerful computers are used, the detection performance can be further reduced by order of magnitudes.

The test data is produced by calling the randomized simulation function with size 1024 (2^{10}), each test folder having 1024 sub-signal 1600-D signals. Notice that due to the randomized nature of these data generation functions, a different call the simulation function, which was harnessed to produce the ~140,000
training instances, will yield different test data different from the previous run for training data generation as the waveforms and noise levels are controlled by random numbers drawn on the fly, the test and training data therefore are different. As described in Sec. x.2.3 and illustrated as the blue data flow in Fig. x.2, all test data are first preprocessed to generate the corresponding $r_a$ or $r_b$ representations, which are then enhanced by the learned LSTM-RNN sub-signal predictor, the final sub-signal detection results are declared by calling the MM-based post-processing procedure described in Sec. x.2.3, esp. the procedure of optimality checking for $t_{p(0)}$, from candidate results generated by Eq. (x.8).

In Fig. x.6, sub-signal detection results for eight typical signals generated by the simulation function produced by the proposed algorithm are illustrated. Each panel consists of four images corresponding to the original simulation signals, data enhanced by the trained LSTM-RNN models, the final sub-signal detection results, and the known ground truths for sub-signals, respectively, are illustrated. To provide a comparison between these two different complex number conversions, detection results for the same eight signals are shown in the upper and lower part for both methods on Fig. x.6, with the same layout. The four numbers are the measures widely used to evaluate the classifier qualities: accuracy, precision, recall and $f1$ scores, based on True Positive (TP): predicted sub-signals are actual sub-signals per ground truths, False Positive (FP): predicted sub-signals are not actual sub-signals, False Negative: the actual sub-signals are not predicted, and True Negative (TN): the actual non-sub-signals are predicted correctly, the formulas for all four measures are tabulated in Tab. X.I. As briefly discussed in the preprocessing step at Sec. x.2.1, this sub-signal detection problem is uneven in the following three aspects:

1. **Unbalanced occurrences**: the positives, namely, presence of sub-signals, are not as many as the negatives, only about 30~40% are sub-signals in each 1600-D signal by running the simulation function, which is the reason we developed the Choice rules to insure the balance of training data.

2. **Different importance in positives and negatives**: in practical applications, the presence of sub-signals is far more important than their absence because urgent reactions are needed should sub-signals make their appearance. Furthermore, missing an actual sub-signal is a far more serious mistake than missing a negative.

<table>
<thead>
<tr>
<th>Tab. X.I. Definitions of quality measures</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>accuracy</strong> = $\frac{TP + TN}{TP + TN + FP + FN}$</td>
</tr>
<tr>
<td><strong>precision</strong> = $\frac{TP}{TP + FP}$</td>
</tr>
<tr>
<td><strong>recall</strong> = $\frac{TP}{TP + FN}$</td>
</tr>
<tr>
<td><strong>$f1$</strong> = $\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$</td>
</tr>
</tbody>
</table>

In view of the foregoing two unevenness of this detection problem, let us check the properties of the four quality measures tabulated in Tab. I to see if they serve our needs.

1. **Accuracy** is defined by treating both positives and negatives equally, the unbalanced nature of sub-signals alone will seriously bias its usefulness: if in a signal the size of the sub-signal is short, e.g., the one in Row 2 Col 3 of Fig. x.6, a detection algorithm totally missing the short sub-signal will still result in a 97% accuracy, which is inappropriate to fairly measure the quality of our sub-signal detection.

2. **Precision** measures the ratio of true positives from the positives predicted by the classifier, it unfairly favors those “thrifty” classifiers that declare presence of positives only when very strong evidence presents: the precision is always 1.0 if the declared sub-signals are always a subset of the actual ones. Due to the higher price of missing actual positives in sub-signal detection tasks, precision’s usefulness is rather limited.
3. *Recall* indicates how many actual sub-signals are really predicted, since it focuses on positions with true sub-signals, the relative sparseness of sub-signals is resolved. Furthermore, it directly measures how many actual sub-signals are missed, which is of crucial importance in practical applications. Therefore, recall is a valuable quality indicator of our problem and should be employed. It must be remarked that recall is imperfect either: an over “*generous*” classifier that predicts many more positives will have a high recall value, which is problematic.

4. *F1 score* is the harmonic mean of precision and recall, because of the limited usefulness of precision compared with recall and recall’s own shortcoming, this measure should be used. It is however not a perfect fit for our task in hand. Given that in practical applications the danger of a false negative is much higher than that of false positive in sub-signal detection problem, probably a better indicator could be a weighted sum of recall and precision with heavier weight goes to recall. Despite its over-

Fig. x.6. Detection results by the proposed sub-signal algorithm using $r_a$ and $r_b$. Each panel consists of four graphs: row 1: original signals; row 2: signals enhanced by LSTM-RNN model; row 3: sub-signal detection after post-processing; row 4: sub-signal ground truths. The four numbers under each panel are accuracy, precision, recall and f1 score, respectively, for the 1600 positions. The panels for $r_a$ and $r_b$ detection results are put in the same layout for ease of comparisons.
emphasis on precision, we still use it as one measure of classifiers, just keep in mind its potential weakness.

To learn the performance of these two different schemes, a Monte Carlo study is conducted: we run the randomized simulation function 20 iterations independently, each iteration generates $2^{10}$ test instances. The proposed detection algorithm, that is, the blue flow depicted in Fig. x.2, is then performed to obtain the detection results, for $r_a$ and $r_b$ based methods. The accuracy, precision, recall and f1 scores for three representative tests of this Monte Carlo study is shown in Fig. x.7. To illustrate the difference made by the introduction of LSTM-RNN data enhancement step, detection results based on $r_a$ and $r_b$ skipping the LSTM-RNN step in Fig. x.2 are also collected where only the conventional signal processing techniques are employed. In our $r_b$ based method, the real and imaginary parts of a complex number went through an absolute operator to be converted to positive real numbers. To inspect what may happen when this absolute operator, Eq. (x.7), is skipped, i.e., using the real and imaginary parts as two real numbers in the 86-D representation, the detection results by this method are also collected and compared. As seen from Fig. x.7, the overall performances, per all four quality measures, achieved by these two complex number conversion approaches are of similar performances. The other three comparison methods failed to deliver consistent performances. Interestingly, the $r_a$ without RNN algorithm delivered comparable performances based on accuracy and precision measures, it is even the best overall (0.95) in precision. However, all these three methods have trouble based on recall: for the randomly generated data set No. 3, all three comparison methods failed gravely with recall values less than 0.4. Even for this “worst” test data set, $r_a$ without RNN has the best precision, 0.98, while getting a mere 0.28 recall. Hence without LSTM-RNN data enhancement step, the sub-signals announced by the conventional signal processing method are almost all true signals, however, without data enhancement most actual sub-signals, 78%, are missed, thus a total failure. But per accuracy (0.86) and precision (0.98), this dire situation is not reflected at all. This confirmed our foregoing discussion about these two quality measures, only the recall and f1 scores can properly represent the performances of the sub-signal detection algorithms, and recall should be given most attention.

![Fig. x.7. Performance measures attained by $r_a$ and $r_b$ for three representative simulation data sets for the two proposed algorithms and three comparison methods.](image-url)
In Fig. x.8, the recall and f1 boxplots of all five methods for the Monte Carlo study are presented. The two proposed algorithms are better than the other three contrasting approaches by a large margin. The performances achieved by these two algorithms are promising given the practical nature of the simulation functions. In addition, while the other three methods have two or three total failures, the two proposed algorithms perform quite robustly in all cases, \( r_a \) based method is slightly better than the \( r_b \) based one, but statistically insignificant. The data enhancement achieved by LSTM-RNN is thus crucial to robustly detect sub-signals. However, the use of original real and imaginary component of complex numbers delivers consistently the worst performance—worse than the two relatively simple methods without going through the LSTM-RNN enhancement procedure, thus the careful choice of data representation is crucial for sub-signal detection tasks. The phase information carried by using the original values of the real and imaginary parts thus hampers the classification, more investigations are still needed on this issue soon.

![Box plots of Recall and f1 scores](image.png)

**Fig. x.8.** The box plots of the Recall (Left) and f1 scores for the Monte Carlo studies. The two LSTM-RNN (Columns 1 & 3) based approaches \( r_a \) and \( r_b \) are significantly better than the other three comparison methods: without RNN enhancement procedure (\( r_a \)-noRNN and \( r_b \)-noRNN) and \( r_b \) without taking the absolute operator.

In view of the extremely similar performance attained by \( r_a \) and \( r_b \) based methods as shown in Figs. x.7 and x.8, if their detection results are mostly the same, then they are likely to get hold of the same information from the sub-signals and thus there is no need to have both approaches. However, as already hinted at from Fig. x.6, this is not the case: the detected sub-signals by these two methods are not photocopies of the other. Indeed, due to their different ways of encoding the underlying complex variables, one mapping all complex number of the same magnitudes to the same real number, the other mapping four points in the four quadrants to one pair of real positive number, the information they captured should be different. As shown Fig. x.9, four new pairs of signals are provided to showcase the differing performances of \( r_a \) and \( r_b \) based methodologies. In cases where the noises are exceedingly severe, these two methods responded differently. Hence these two methods could complement each other by taking account of different aspects of the noisy sub-signal signals. A simple combination of the detection results by these two methods, e.g., a simple OR as seemingly suggested by Fig. x.9, is not working. Cases shown in Fig. x.9 suggest this simplistic scheme works. For many more cases, false positives and false negatives by both methods add up and result in worse accuracy. The average accuracy for all simulation sets used in Monte Carlo studies reported in Figs. 7 and 8. This over-simplistic OR combination results in slightly but consistently worse performances than the
minimal of the two original approaches. More methods are still needed if the ensemble methodology [21] can be seriously evaluated.

**Fig. x.9. Different performances of \( r_a \) (Row 1) and \( r_b \) (Row 2) based detection results: while \( r_a \) method failed to detect the two sub-signals due to severe noises (Columns 1 & 2 in Row 1), \( r_b \) based method has no trouble picking up the sub-signals. Conversely, while \( r_b \) missed the two sub-signals in the last two columns (Columns 3 & 4 in Row 2), \( r_a \) based method detected them perfectly.**

**X.4. CONCLUSION**

To detect complex sub-signals severely contaminated by noises for the security of intelligent vehicles, we developed a new deep learning approach combined with a signal post-processing algorithm. In this work, the uneven and unbalanced signals are first preprocessed and regularized as a local vector for proper training and testing by deep learning. The complex numbers are converted to real ones using two different schemes: one using the magnitude of the complex number, the other using the absolute value of a complex number’s real and imaginary component. The LSTM-RNN was exploited to enhance the sub-signals that otherwise are buried by background noises. The enhanced data are next classified by use of a global signal analysis procedure based mostly on mathematical morphology. These two algorithms were tested rigorously using the simulation functions made available to us, the performances achieved are encouraging. The great power of the deep learning and conventional signal processing are effectively harness to achieve valuable detection results.

This is our first effort to detect sub-signals from noisy and complex signals by combining the prowess of conventional signal processing techniques with deep learning approaches and this is an on-going endeavor. Next, three different directions will be explored to gain deeper insights into this task and attain better results:

1. Change simulation functions to expose more parameters for us to control, so that in our Monte Carlo studies, we can fine-tune these parameters to have better knowledge about when and why our algorithms succeed or fail. This way more fine-tuning can be conducted to improve and generalize our algorithms.

2. Use different deep learning methods for data enhancement. In this work, we only used LSTM-RNN for data enhancement, CNN based method, given its great power of exploiting local dependencies with multiple resolutions, will be examined to detect sub-signals. The sub-signal detection may benefit from the higher-dimensions involved in CNN framework.

3. Explore direct complex number representation based methods. In this work we relied on TensorFlow’s backend for LSTM-RNN training, which has no support for complex numbers so far. We thus formulate two different ways to load the complex information by real numbers: one using
complex magnitude, the other using absolute values of the real and imaginary components. We have proved empirically that the literal use of the real and imaginary components of complex numbers cannot yield desirable results. In parallel to our on-going research and development, Bengio’s group made impressive progress in deep learning using complex numbers directly [29] based on their Theano backend, whereof they reported ~3% performance gains due to the direct use of complex numbers. We will next study how to plug this deep complex net in our data enhancement framework with improved performance.

REFERENCES


