DeepDetect: A Cascaded Region-based Densely Connected Network for Seismic Event Detection

<table>
<thead>
<tr>
<th>Journal:</th>
<th>Transactions on Geoscience and Remote Sensing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuscript ID</td>
<td>TGRS-2017-01371.R1</td>
</tr>
<tr>
<td>Manuscript Type:</td>
<td>Regular paper</td>
</tr>
<tr>
<td>Date Submitted by the Author:</td>
<td>n/a</td>
</tr>
<tr>
<td>Complete List of Authors:</td>
<td>Wu, Yue; Los Alamos National Laboratory, Geophysics Lin, Youzuo; Los Alamos National Laboratory, Geophysics Zhou, Zheng; University of Rochester, Electrical and Computer Engineering Bolton, David; Penn State University, Department of Geosciences Liu, Ji; University of Rochester, Department of Computer Science Johnson, Paul; Los Alamos National Laboratory, Geophysics</td>
</tr>
<tr>
<td>Keywords:</td>
<td>Seismic measurements, Time series, Signal detection</td>
</tr>
</tbody>
</table>
DeepDetect: A Cascaded Region-based Densely Connected Network for Seismic Event Detection

Yue Wu¹, Youzuo Lin¹⁺, Zheng Zhou², David Chas Bolton³, Ji Liu², and Paul Johnson¹,*

Abstract—Automatic event detection from time series signals has broad applications. Traditional detection methods detect events primarily by the use of similarity and correlation in data. Those methods can be inefficient and yield low accuracy. In recent years, machine learning techniques have revolutionized many sciences and engineering domains. In particular, the performance of object detection in 2D image data has significantly improved due to deep neural networks. In this study, we develop a deep-learning-based detection method, called “DeepDetect”, to the detect events from seismic signals. We find that the direct adaptation of similar ideas from 2D object detection to our problem faces two challenges. The first challenge is that the duration of earthquake event varies significantly; The other is that the proposals generated are temporally correlated. To address these challenges, we propose a novel cascaded region-based convolutional neural network to capture earthquake events in different sizes while incorporating contextual information to enrich features for each proposal. To achieve a better generalization performance, we use densely connected blocks as the backbone of our network. Because some positive events are not correctly annotated, we further formulate the detection problem as a learning-from-noise problem. To verify the performance, we employ seismic data generated from the Pennsylvania State University Rock and Sediment Mechanics Laboratory, and we acquire labels with the help of experts. We show that our techniques yield high accuracy. Therefore, our novel deep-learning-based detection methods can potentially be powerful tools for identifying events from time series data in various applications.

Index Terms—Convolutional neural network (CNN), seismic signals, time series segmentation, event detection.

I. INTRODUCTION

TIME series data can be acquired through sensor-based monitoring. In the past few years, there has been increased interest to detect useful events from time series datasets for different applications. Among these problems, seismic monitoring to detect earthquakes has attracted many interests [1, 2]. In this study, we develop a novel event detection method and further employ our method to seismic time series datasets.

Machine learning methods have been successful in object detection to identify patterns. There have been many existing machine learning methods to detect events from time series datasets for various applications such as epileptic seizure detection from EEG signals and change detection from remotely sensed imagery datasets. Depending on the availability of labeled datasets, these event detection methods for time series datasets can be categorized into supervised [2, 3, 4, 5] and unsupervised methods [6, 7]. Our study belongs to the supervised category, since we acquire labels for training and evaluation with the help of experts. As for supervised methods, they are all point-wise detection methods meaning they classify data points at each time stamp. Point-wise detection methods can be limited in their detection performance. In particular, those methods can neither accurately localize events nor obtain the number of events. In this study, inspired by the object detection in 2D imagery, we develop a novel event-wise detection method, called “DeepDetect”, to capture each complete event. In other words, our DeepDetect captures the beginning and end coordinates to localize each event from the time series datasets.

Convolutional neural networks (CNN) has achieved promising results in computer vision, image analysis, and many other domains due to the significantly improved computational power ([8, 9, 10, 11]). The State-of-the-Art CNN-based object detection models for 2D imagery primarily consist of two steps [12, 13, 14]: a step to generate the region proposals and a step to identify and localize the events within proposals. Specifically, segments of the input data that may include targeting patterns are first used to generate region proposals. A classifier is then employed on each proposal to detect targeting patterns, and a regressor is utilized to localize events within positive proposals. The original proposal generation method for CNN-based detection models is developed in two region-based CNN models, known as R-CNN [14] and Fast R-CNN [13], where fixed methods are used to obtain proposals. Faster-RCNN [12] improves previous models by building region proposal networks (RPN) on top of the final feature map of CNN backbone. Compared with Girshick [13] and Girshick et al. [14], the Faster-RCNN eliminates the additional time spent on proposal generation. To determine whether a proposal is positive or negative, Ren et al. [12] introduces the concept of anchor to denote the region on the input data that a proposal covers. A proposal is considered positive if its corresponding anchor overlaps the ground-truth above a threshold.

In this study, we develop a novel deep neural network detection method for time series datasets. Similar to previous 2D detection models, our model also consists of two steps: proposal generation and event localization. However, a direct adaptation of 2D methods to generate the region proposals does not work well with our 1D seismic time series datasets because the duration of seismic events varies...
significantly. Therefore, we develop a novel region proposal method to address this issue. In particular, we develop a cascaded network that generates proposals by including more down-sampling layers than regular networks do. Theoretically, events of small size can be captured at shallow layers. As the network becomes deeper, events of large size can be captured due to the increasing size of the receptive field. We add detection branches on feature maps at different depths so that events in various scales can be captured.

Features are critical to the performance of our detection model. Since the classifier and regressor in the second step share the same feature vector obtained from the CNN, enriching features for proposals will boost the detection rate and localization accuracy. Another novelty of our work is the incorporation of contextual information for each individual proposal. Although the importance of contextual information has been emphasized for imagery segmentation [15, 16, 17], there are surprisingly few detection models taking into account contextual information on the proposal level. As for the time series seismic signals, proposals are temporally correlated. Utilizing each proposal individually generates many false-positive detections. This is because proposals may be part of some large events, and our detection method should be able to distinguish small signal segments from large events. Considering this, we enrich features of each proposal by incorporating contextual information.

Because of the cascaded structure, the number of parameters in our model may significantly increase. To obtain better generalization performance, we build our DeepDetect based on densely connected networks (DenseNet) [8]. The core idea of DenseNet is to reuse features learned from shallow layers, which enables us to maintain a reasonable number of parameters even if the network becomes substantially deep. Another strategy we use to address overfitting is to share the parameters of the sibling detector and regressor. This is reasonable since we are interested in capturing specific patterns regardless of their size.

Due to the variation in the event patterns and density, it is impractical for domain experts to accurately annotate all events. Those omitted events may bias the classifier for proposals. To alleviate the impact of mis-labeled positive events, we further formulate the proposal classification as a learning-from-noise problem. Inspired by Natarajan et al. [18], we use a label-dependent loss function for the classifier.

We test our detection models on seismic time series data and compare the experimental results obtained using the proposed cascaded contextual region-based CNN (CC-RCNN) and the traditional template matching method. We use AP@[0.5, 0.95] as the evaluation metric. Average precision (AP) calculates the averaged maximum precision at each recall value. We calculate 10 APs with the intersection over union (IoU) of [0.5:0.95:0.05] as the metric to identify true positive detections and then take the average. We also conduct ablation experiments to verify the effect of the multi-scale architecture and the incorporation of contextual information. The experiment results demonstrate that our deep-learning-based model achieves AP@[0.5, 0.95] of 63.8%, which significantly outperforms the template matching method. The ablation studies further show that the incorporation of contextual information for each individual proposal not only reduces false-positive detections, but also significantly increases the event localization accuracy. Also the utilization of label-dependent loss further boosts the performance of our detection models. To summarize, our contributions can be listed as follow:

- Extend region-based convolutional neural networks to time series scenarios;
- Propose a cascaded structure to generate multi-scale proposals to efficiently capture events in varying lengths;
- Incorporate contextual information for each proposal to further boost the detection accuracy;
- Conduct experiments on seismic time series data and obtain promising results–achieving average precision (AP)@[0.5, 0.95] of 63.8%.

The paper is organized as follows. Section II briefly reviews related works on event detection and object detection. Section III gives background knowledge about our method. The proposed method is elaborated in Section IV. Implementation details of the proposed methods and counterpart methods are described in Section V. Experiment results are provided in Section VI. Section VII concludes this paper.

II. RELATED WORK

Our study is related to both event detection for 1D time series datasets and object detection for 2D imagery datasets.  

1) Detection Methods for Time Series Data: There are many event detection methods in various applications. In seismology, STA/LTA is the most popular used detection method due to its simplicity [19, 20]. STA/LTA computes the ratio of short-term average energy and long-term average energy on multiple receivers. If the ratio exceeds a predefined threshold, a detection is declared. However, STA/LTA fails to detect earthquakes or yields many false detections in challenging situations such as signal with low signal-to-noise ratio, or overlapping events. Compared to STA/LTA, autocorrelation yields a much higher detection rate [21]. Autocorrelation is an exhaustive “many-to-many” detection method. It searches for similar waveforms when the desired signal waveform is unknown. The major disadvantage with autocorrelation lies in its notoriously expensive computational cost, which scales quadratically with data duration. Template matching is a detection method that yields a good balance between accuracy and computational complexity [22, 23]. Template matching is a “one-to-many” detection method. It computes the correlation coefficient of a template waveform with the candidate waveform data. A detection is claimed when the correlation coefficient value is above the user-defined threshold value. Template matching has been proven to be efficient and successful for different seismic applications: microseismic monitoring in geothermal fields [24], oil and gas reservoirs [25], nuclear monitoring [26], and tectonic tremor [23], etc. However, template matching usually performs poorly when events are dissimilar. Yoon et al. [2] recently developed an event detection approach called the fingerprint and similarity thresholding (FAST) method and applied it to detect earthquakes out of seismic datasets. FAST creates
“fingerprints” of waveforms by extracting key discriminative features and then groups similar fingerprints together within a database to facilitate the fast and scalable search for similar fingerprint pairs. There are many other event detection methods developed in other application domains. Oehmcke et al. [3] employed local outlier factor to detect events from marine time series data. To further improve results, dimensionality reduction methods are employed by the authors to the datasets. In the work of Batal et al. [5], an event detection method was developed based on recent temporal patterns. The detection algorithm mines time-interval patterns backward in time, starting from patterns related to the most recent observation. The authors further applied their detection method to health care data of diabetic patients. McKenna et al. [27] developed a binomial event discriminator (BED) method. BED uses a failure model based on the binomial distribution to determine the probability of an event within a time segment. The effectiveness of their model is demonstrated with hydrological datasets.

2) CNN-based Detection Methods for Image Data: Ren et al. [12] developed the faster RCNN method, of which a window is slid on the final feature map of the fourth stage of ResNet [9] to generate proposals. The authors use nine different anchors with three various sizes (128, 256, 512) and three ratios of height/width (1:1, 1:2, 2:1) to determine regions that a proposal covers. To make anchors more accurate, Cai et al. [28] developed the multi-scale CNN methods, consisting of a proposal sub-network to generate multi-scale proposals at three stages of VGG [10] network. The authors then built detectors on top of each proposal branch.

III. BACKGROUND AND RELATED WORK

A. Template Matching

Template matching (TM) [29] is widely used in the seismology community. It calculates the similarity of a template with successive windows from continuous waveform data. The commonly used similarity metric is normalized cross-correlation (CC),

\[
CC(a, b) = \frac{(a, b)}{\|a\|_2 \|b\|_2} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2} \sqrt{\sum_i b_i^2}},
\]

where \(a, b\) are vectorized time series signals. The detection threshold of \(\tau\) is given as

\[
\tau = \mu \cdot \text{median absolute deviation (MAD)},
\]

where \(\mu\) is usually chosen as 9 [30, 31, 32]. For a univariate set \(X_1, X_2, ..., X_n\), MAD can be calculated as

\[
\text{MAD}(X) = \text{median}(|X_i - \text{median}(X)|).
\]

B. Densely Connected Network

Densely connected network (DenseNet) [8] is an improved version of the residual network [9]. Both ResNet and DenseNet are discussed in this section.

1) ResNet Block: The major breakthrough of ResNet is the application of skip connections between residual blocks, which can be denoted as

\[
x_{l+1} = x_l + W^{*} \ast (\sigma(B(W \ast (\sigma(B(x_l)))))
\]

where \(x_l\) is the input feature to the \(l\)th residual block, \(W\) and \(W^{*}\) are weight matrices, the operator of “\(\ast\)” denotes convolution, \(B\) denotes batch normalization (BN) [33], \(\sigma(x) = \max(0, x)\) [34]. Eq. (4) forms a building block in ResNet. Skip connections are implemented by summing up the input of the block and the output of a set of convolution layers. The existence of skip connections weakens the importance of each individual path, so that the model behaves like an ensemble of small networks. Although ResNet immediately topped most of benchmarks, a few drawbacks still need to be addressed. 1) The number of parameters becomes extremely large with hundreds layers of convolutions. 2) The network may not benefit from “going deep” due to the gradient vanishing problem, as indicated in Veit et al. [35].

2) DenseNet Block: A DenseNet block is then formulated as

\[
x_{l+1} = \mathcal{H}([x_0, x_1, ..., x_l])
\]

\[
\mathcal{H}(x) = W \ast (\sigma(B(x))),
\]

where \(x_0, x_1, ..., x_l\) denote the input of each convolution layer in the block, the bracket “[” denotes the concatenation of all outputs of previous layers. A DenseNet block is illustrated in Fig. 1.

Thus, the output of a layer in one DenseNet block is densely connected with outputs of all deeper layers in the same block by means of concatenation. Thus, it fully exploits the advantage of skip connections. Moreover, features from shallow layers are reused by deep layers, which reduces the number of parameters, and the gradient vanishing problem is further alleviated by the concatenation layers.

The feature dimension \(d_{l+1}\) of \(x_{l+1}\) is calculated as

\[
d_{l+1} = d_0 + k \cdot l,
\]

where \(k\) (a.k.a the growth rate) is the number of filters used for each convolution layer.

C. Atrous Convolution

Atrous convolution convolves input nodes with a dilation rate \(d\), denoting the stride for each convolved location on input nodes. The output node \(y_i\) of an atrous convolution layer is calculated as

\[
y_i = \sum_{k=1}^{K} x_{i+d \cdot k} \cdot w_k,
\]

where \(k\) is the dimension, \(x_i\) is the input node and \(w \in R^K\) is the kernel.

The regular convolution can be seen as a special case of atrous convolution with \(d = 1\). Atrous convolution was
first proposed in [36] to address the low-resolution problem caused by downsampling layers (pooling, convolution with stride, etc.). Atrous convolution essentially involves distant information by covering larger regions of input signals while maintaining the same number of parameters.

IV. PROPOSED METHODS

A. Network Architecture

<table>
<thead>
<tr>
<th>Stage</th>
<th>Layers</th>
<th>Dim.</th>
<th>Anchor Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv1</td>
<td>conv1, 120</td>
<td>L/24</td>
<td>-</td>
</tr>
<tr>
<td>Avg-pool1</td>
<td>avg-pool1, /2</td>
<td>L/42</td>
<td>-</td>
</tr>
<tr>
<td>Conv2</td>
<td>conv2, 20</td>
<td>L/42</td>
<td>-</td>
</tr>
<tr>
<td>Avg-pool2</td>
<td>avg-pool2, /2</td>
<td>L/82</td>
<td>-</td>
</tr>
<tr>
<td>Conv3</td>
<td>conv3, 20</td>
<td>L/82</td>
<td>-</td>
</tr>
<tr>
<td>Avg-pool3</td>
<td>avg-pool3, /2</td>
<td>L/162</td>
<td>-</td>
</tr>
<tr>
<td>Conv4</td>
<td>conv4, 20</td>
<td>L/162</td>
<td>-</td>
</tr>
<tr>
<td>Avg-pool4</td>
<td>avg-pool4, /2</td>
<td>L/322</td>
<td>-</td>
</tr>
<tr>
<td>Conv5</td>
<td>conv5, 20</td>
<td>L/322</td>
<td>-</td>
</tr>
<tr>
<td>Avg-pool5</td>
<td>avg-pool5, /2</td>
<td>L/642</td>
<td>-</td>
</tr>
<tr>
<td>Conv6</td>
<td>conv6, 20</td>
<td>L/642</td>
<td>-</td>
</tr>
<tr>
<td>Avg-pool6</td>
<td>avg-pool6, /2</td>
<td>L/1282</td>
<td>-</td>
</tr>
<tr>
<td>Conv7</td>
<td>conv7, 20</td>
<td>L/1282</td>
<td>-</td>
</tr>
<tr>
<td>Avg-pool7</td>
<td>avg-pool7, /2</td>
<td>L/2562</td>
<td>-</td>
</tr>
<tr>
<td>Conv8</td>
<td>conv8, 20</td>
<td>L/2562</td>
<td>-</td>
</tr>
<tr>
<td>Avg-pool8</td>
<td>avg-pool8, /2</td>
<td>L/5122</td>
<td>-</td>
</tr>
<tr>
<td>Conv9</td>
<td>conv9, 20</td>
<td>L/5122</td>
<td>-</td>
</tr>
<tr>
<td>Avg-pool9</td>
<td>avg-pool9, /2</td>
<td>L/10242</td>
<td>-</td>
</tr>
</tbody>
</table>

TABLE I: The cascaded network architecture of DeepDetect for event detection. L denotes the length of the input waveform. "-" means the output of that stage is not used to make predictions.

As illustrated in Table I, our DeepDetect is inspired by DenseNet. All convolution kernels in our network are 1D because of the input of 1D time series data. “Conv7, 64, /2” denotes using 64 1 × 7 convolution kernels with stride 2. The same routine applies to max-pool and avg-pool. The brackets denote DenseNet blocks illustrated in Fig. 1. Specifically, we use “[conv3, 20] × 6” to denote a DenseNet block where 6 convolution layers are applied, each followed by batch normalization and ReLU activation layers. \(D_i\), \(T_i\) denote DenseNet blocks and transition blocks, respectively. All transition blocks have an average pooling layer to downsample the signal by 2, while \(T_3 \sim T_9\) have an extra 1 × 1 convolution layer to reduce the feature dimension by half. As previously discussed, our model is designed for capturing events with significantly different durations, so we use the output of each \(D_3 \sim D_9\), having strides 16, 32, 64, 128, 256, 512, 1024 on the input signals, as proposals, then detection branches (the classification and regression layer) are built on the top of these multi-scale proposals. Since small events greatly outnumber large events, we share the detection branches for all scales to make our model robust. To achieve this, We set growth rate \(k = 12\) for \(D_1, D_2\), \(k = 20\) for \(D_3 \sim D_9\), so that all proposals have the same feature dimension 240.

B. Anchors

Anchor is the effective region of the input signals that a proposal is responsible for. In most cases, it is used to decide the label for that proposal. In our time series data, an anchor indicates two coordinates representing the beginning and the end of each proposal. We assign the anchor size 128 to proposals in \(D_1\), and it doubles for the next scale. Proposals in \(D_9\) have the largest anchor size 8192. These settings are determined by the length distribution of events in our data. It is worthwhile to mention that the amount of shift between adjacent proposals is determined by the stride of that stage on the input signals. Specifically, the stride is 1/8 of the anchor size for each \(D_3 \sim D_9\).

C. Proposals with Contextual Information

Features are critical to detection. In time series data, it is important to take into consideration of the temporal correlations among neighboring proposals. Only considering features from each individual proposal will result in many false detections. Figure 2a illustrates a perfect individual event in time series data.
All convolutions are rates, followed by batch normalization and activation layers. Outputs of atrous convolutions with 4, 8, and 12 as dilation and contextual proposals are 0.5, 1, 1.5 of the anchor size. The blocks in red, green, and blue shown in Fig. 3 are the and adjacent proposals. Anchors of adjacent proposals shift only a little, hence the features of adjacent proposals tend to be similar. In contrast, atrous convolution is capable of incorporating contextual information. For the other extreme, the shifts of anchors should not be too large since the information from far away will be irrelevant to the target proposal. With dilation rates of 4, 8, 12, the shifts between the target proposal and contextual proposals are 0.5, 1, 1.5 of the anchor size. The blocks in red, green, and blue shown in Fig. 3 are the outputs of atrous convolutions with 4, 8, and 12 as dilation rates, followed by batch normalization and activation layers. All convolutions are 1 × 3, with 240 kernels. We generate new proposals with contextual information by concatenating outputs using three dilation rates and the target proposal. The new proposal includes four times as many features as the target proposal. In order to keep the number of features unchanged, we further employ a 1 × 1 convolution layer. To summarize, we employed atrous convolution with 3 dilation rates on proposals layers so that the features of each individual proposal are enriched by preceding and succeeding proposals, while maintaining its own features.

Fig. 3: The atrous convolution block. We aim at enriching features of individual proposals by convolving with preceding and succeeding proposals. In this figure, the target proposal is at the center. Atrous convolutions with different strides are applied to capture contextual information from nearby to further proposals.

We add a classification branch and a regression branch on each proposal. The classification branch is first used to detect whether a proposal includes an event or not. For each positive proposal, we further apply a regressor to localize the event within. We use a joint loss function to optimize classification and regression branches simultaneously.

We assign a positive label to a proposal if its anchor has the ratio of intersection over union (IoU) above 0.5 with at least one ground-truth event. Proposals are assigned a negative label if the highest IoU of their anchors with the ground-truth is below 0.3. Neutral proposals (IoU ∈ [0.3, 0.5]) do not contribute to the loss. To localize the event within a proposal, two offsets: \( d_x \) and \( d_w \) are captured to transform the anchor to real coordinates by

\[
\begin{align*}
G_x &= P_w d_x + P_x, \\
G_w &= P_w \exp(d_w),
\end{align*}
\]

where \( P_x, P_w \) are the center and length of an anchor, \( G_x, G_w \) are the center and length of the prediction. Figure 4 gives an illustration of this process, where three nodes are positive so they also have a localization loss.

Another challenge in our time series data is that not all events in the training set are annotated, which is caused by the fact that some patterns are difficult for our annotators to determine. Due to this problem, negative labels are noisy in our task. To address this issue, we employ label-dependent loss function for the classifier.

The label-dependent loss function was initially proposed in a couple of works [37, 38], known as weighted logistic regression and biased support vector machine, respectively. The core idea of “label-dependent” is to apply separate loss functions for positive and negative sets

\[
J(g(x)) = \frac{1}{|X|} \left( \alpha \sum_{x \in X_+} l(g(x)) + \beta \sum_{x \in X_-} l(g(x)) \right),
\]
where $l$ can be any 0-1 loss functions, \( X_+ \), \( X_- \) denote the observed positive and negative sets, \( \alpha \) and \( \beta \) are two hyper-parameters, and \( g \) is a linear score function.

To obtain the optimal weight parameters \( \alpha^* \) and \( \beta^* \), Natarajan et al. [18] give

\[
\rho_{+1} = P(Y = 1|\bar{Y} = -1),
\]

\[
\rho_{-1} = P(Y = -1|\bar{Y} = 1),
\]

\[
\alpha^* = \frac{1 - \rho_{+1} + \rho_{-1}}{2},
\]

\[
\beta^* = 1 - \alpha^*,
\]

where \( \bar{Y} \) denotes the observed label, \( Y \) denotes the true label and the conditional probability \( P \) denotes the probability of a sample being wrongly labeled. It can be shown that by employing the optimal parameters of \( \alpha^* \) and \( \beta^* \), the resulting classifier can make predictions of sign(\( g(x) - 1/2 \)) with noisy data [18]. We use the similar parameter estimation approach to our datasets by setting \( \rho_{+1} = 0 \), since the noise only exists in negative samples.

1) Loss Function: We develop a joint loss function \( L \) including a classification loss function \( L_{cls} \) and a regression loss function \( L_{regr} \)

\[
L(d_{cls}, d_u, d_w, t_{cls}, t_u, t_w) = L_{cls}(d_{cls}, t_{cls}) + \lambda 1\{t_{cls} = 1\} \sum_{u \in \{x, w\}} L_{regr}(d_u, t_u),
\]

where \( 1\{t_{cls} = 1\} \) is the indicator function indicating only positive proposals contribute to the regression loss, and \( d_{cls}^{(i)}, d_u^{(i)}, d_w^{(i)} \) and \( t_{cls}^{(i)}, t_u^{(i)}, t_w^{(i)} \) are the corresponding ground-truth of the \( i \)th proposal’s class score, center and length offsets, respectively, and \( \lambda \) controls the balance between \( L_{cls} \) and \( L_{regr} \).

The classification loss function \( L_{cls} \) is defined as a label-dependent logistic loss

\[
L_{cls}(d_{cls}, t_{cls}) = \alpha 1\{t_{cls} = 1\} \log(1 + e^{-d_{cls}}) + (1 - \alpha) 1\{t_{cls} = -1\} \log(1 + e^{d_{cls}}),
\]

where \( \alpha \) is the hyper-parameter.

The regression loss function \( L_{regr} \) is defined as the smoothed \( L_1 \) loss as proposed in [13]:

\[
L_{regr}(d_u, t_u) = \text{smooth}_{L_1}(t_u - d_u),
\]

where \( d_u, t_u \) denote the predicted and the target values, and

\[
\text{smooth}_{L_1}(x) = \begin{cases} 
0.5x^2 & \text{if } |x| < 1 \\
|x| - 0.5 & \text{otherwise}
\end{cases}
\]

According to Eqs. (9) and (10), \( t_x \) and \( t_w \) can be obtained by

\[
t_x = (G_x^* - P_x)/P_w, \tag{19}
\]

\[
t_w = \ln(G_w^*/P_w), \tag{20}
\]

where \( G_x^* \) and \( G_w^* \) are the ground-truth of the center and length of the event.

2) Share Weights for Robustness: To capture events with dramatically varying durations, we make multi-scale predictions on output layers with different sizes of receptive fields. However, events with different lengths are not equally distributed. In other words, small events greatly outnumber large events. Our model should capture patterns from all events regardless of their durations or amplitudes; hence we share the weights of contextual atrous convolution layers, sibling classification and regression branches built on top of \( D_A \). Weight sharing makes our model robust and help with the optimization because predictions in all scales equally contribute to the loss function.

V. IMPLEMENTATION DETAILS

A. Template Matching

We use events in the training set as templates. For each template, CC is calculated at each sliding location of the time series data. We set the detection threshold \( \mu = 8 \) in Eq. (2), which is determined by the validation set. For multi-detection of a single event, the detection with the highest CC is kept and all other detections are discarded. The beginning and the end of each detection are determined by those of the template.

B. Proposed Model

1) Optimization: The proposed model has approximately 3 million parameters. For each mini-batch iteration, we feed a 24,576 timestamp (6s) time series segment with 0.5 overlapping rate so that if the end point of an event lies outside of the segment, that event will be roughly at the center of the next segment. In our dataset, large events are rare comparing to small events. To account for this characteristic, we generate less proposals from large events than those from small events for training. In particular, Table II shows how we select the number of proposals from each detection branch.

<table>
<thead>
<tr>
<th>stage</th>
<th>( C_0^* )</th>
<th>( C_1^* )</th>
<th>( C_2^* )</th>
<th>( C_3^* )</th>
<th>( C_4^* )</th>
<th>( C_5^* )</th>
<th>( C_6^* )</th>
<th>( C_7^* )</th>
<th>( C_8^* )</th>
</tr>
</thead>
<tbody>
<tr>
<td># of proposals</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>32</td>
<td>32</td>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE II: Number of proposals selected from each scale.

To further simplify the optimization, we also make sure the ratio of positive and negative proposals is 1:1. If positive or negative proposals are insufficient, we use neutral ones as negative proposals. Adam optimizer [39] is applied with the initial learning rate of 5e-4. The learning rate is multiplied by 0.1 for every ten epochs. Each mini-batch data is subtracted by the mean and divided by the standard deviation before feeding into the network. Our model is implemented with TensorFlow [40].

2) Inference: Similarity as for the training process, we feed 24,576 timestamp segments with the overlapping rate of 0.5 each time window. Predictions are first generated on all proposals, then we apply non-maximum suppression (NMS) to reduce multi-detections for a single event. Since events in time series data are rarely overlapped, we set the IoU threshold of NMS to be 0.05.
VI. Experiment

A. Data

The data used for our machine learning model comes from a frictional experiment in the Pennsylvania State University Rock and Sediment Mechanics Laboratory. The experiment was conducted using the double-direct shear configuration where two laboratory fault zones are sandwiched between two forcing blocks and sheared simultaneously.

Acoustical data is recorded from two broadband (.02-2MHz) piezoelectric transducers with a central frequency of 500kHz. The transducers are embedded within a 10x10 (cm$^2$) steel block and placed adjacent to the fault zone [43]. Acoustical data is sampled continuously at 4MHz with a 14-bit Verasonics data acquisition system.

We hand-picked 1000 acoustic emissions (AEs) from the raw time series data between 2368.9-2369.7 (s). During this section of the experiment, the normal stress is at 3 MPa. Furthermore, this time corresponds to the first two seconds of the stick-slip cycle. For a single AE, we picked the beginning and end of the event based on the amplitude of the signal, duration of the signal and the characteristic shape of the signal. We use 800 events for training, 100 events for validating, and 100 events for testing.

The length distribution of all events is shown in Fig. 5. The largest event spans more than 7,000 timestamps (1.7 ms) while the smallest event spans few hundreds. The mean and median lengths of events are both about 1,500 timestamps.

B. Metric

We use average precision (AP) to evaluate our models. AP first calculates the precision-recall curve, then averages maximum precisions for each unique recall. For AP@.5, a detection is considered as true positive if it has IoU above 0.5 with a ground-truth event. If there are multiple detections for one event, only one detection is considered true positive, others are considered false positive. In this study, we use AP@[.5, .95], which is used in MS COCO object detection dataset [44]. To obtain AP@[.5, .95], we calculate 10 APs using IoUs from 0.5 to 0.95 with stride 0.05, then take the average of 10 APs. The IoU for two 1D segments $A_0A_1$, $B_0B_1$ is calculated as:

$$ x_a = \max(A_0, B_0), $$
$$ y_a = \min(A_1, B_1), $$
$$ x_b = \min(A_0, B_0), $$
$$ y_b = \max(A_1, B_1), $$

$$ IoU(A_0A_1, B_0B_1) = \frac{\max(y_a - x_a, 0)}{y_b - x_b}. $$

C. Test: Overall Performance

<table>
<thead>
<tr>
<th></th>
<th>TM</th>
<th>CC-RCNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP@[.5]</td>
<td>22.0%</td>
<td>95.7%</td>
</tr>
<tr>
<td>AP@[.55]</td>
<td>12.4%</td>
<td>94.6%</td>
</tr>
<tr>
<td>AP@[.6]</td>
<td>7.4%</td>
<td>91.5%</td>
</tr>
<tr>
<td>AP@[.65]</td>
<td>4.6%</td>
<td>87.5%</td>
</tr>
<tr>
<td>AP@[.7]</td>
<td>3.8%</td>
<td>80.0%</td>
</tr>
<tr>
<td>AP@[.75]</td>
<td>2.4%</td>
<td>72.5%</td>
</tr>
<tr>
<td>AP@[.8]</td>
<td>1.1%</td>
<td>61.5%</td>
</tr>
<tr>
<td>AP@[.85]</td>
<td>0.5%</td>
<td>39.2%</td>
</tr>
<tr>
<td>AP@[.9]</td>
<td>0.3%</td>
<td>15.1%</td>
</tr>
<tr>
<td>AP@[.95]</td>
<td>0.1%</td>
<td>0.7%</td>
</tr>
<tr>
<td>AP@[.5, .95]</td>
<td>5.5%</td>
<td>63.8%</td>
</tr>
</tbody>
</table>

TABLE III: Accuracy results obtained using template matching and our CC-RCNN model. We notice that our CC-RCNN model consistently yields better detection accuracy than the template matching method.

We provide detection results in Table III obtained using the template matching (baseline), and our deep-learning-based CC-RCNN. Table III shows that our CC-RCNN model consistently yields better detection accuracy than the template matching method. The average AP value of our method is 63.8% compared to the value of 5.5% obtained by the TM method.

To visualize the detection results using our method and the TM method, we provide the example detections of these two models in Fig. 6. The results obtained using our method are denoted by “•”, and those obtained by template matching are denoted in “×”. The ground-truth (“•”) is also provided. Although TM has the ability to detect some events, it has poor localization performance. For instance, TM only captures the second half of event #6 in Fig. 6a. Also, TM cannot capture event #7 as a whole since each template works separately. The reason for the poor localization is that the lengths of TM detections are simply determined by those of templates, and most events have lengths that no templates can match. On the other hand, the proposed CNN-based method is capable of accurately detecting, as well as localizing multi-scale events due to the cascaded design. To capture events shown in Fig. 6b is more challenging than those in Fig. 6a since some of events in Fig. 6b vary dramatically in length (such as #57, #58) and yield irregular patterns (such as #60). TM detect #57 - #60 as a whole event, while CC-RCNN is able to detect and localize each individual event accurately.
Based on results shown in Table. III and Fig. 6, our detection methods yield much higher accuracy than template matching method.

D. Test: Hyper-Parameters

<table>
<thead>
<tr>
<th>λ</th>
<th>0.1</th>
<th>1</th>
<th>10</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP@.50</td>
<td>48.2%</td>
<td>60.4%</td>
<td><strong>61.5%</strong></td>
<td>56.0%</td>
</tr>
</tbody>
</table>

**TABLE IV**: Accuracy w.r.t different values of λ in Eq. (16). We test λ ∈ {0.1, 1, 10, 100}. The accuracy peaks when λ = 10.

Hyper-parameters play an important role in our model to achieve high performance. Specifically, the selection of λ value in Eq. (16) and α value in Eq. (15) are critical to the detection accuracy using our label-dependent loss function.

We provide results in Table IV to illustrate the performance of our algorithm using different λ values in Eq. (16). Similar to Ren et al. [12], different values of λ ∈ {0.1, 1, 10, 100} are tested. We observe that the performance of our model are impacted notably by using different λ values. The best performance is achieved with λ = 10, which is therefore used for all the tests implemented within the work.

To demonstrate the effectiveness of different values of α, we further provide detection results using our label-dependent loss function in Eq. (17). The performance on different noise parameters, α ∈ {0.45, 0.5, 0.55, 0.6, 0.7}, is reported in Table V. The best average performance of our CC-RCNN model is achieved when α = 0.55 with the corresponding noise level of 0.1. This is reasonable since the noise only exists in negative samples, and the noise level is low. The accuracy decreases when either the noise level becomes larger (α = 0.55, 0.6, 0.7) or assuming that the noise exists in positive samples (α = 0.45).

Through these sets of tests on hyper-parameter, we obtain the best combination to use for our dataset i.e., λ = 10 and α = 0.55.

<table>
<thead>
<tr>
<th>α</th>
<th>0.45</th>
<th>0.5</th>
<th>0.55</th>
<th>0.6</th>
<th>0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP@.50</td>
<td>97.3%</td>
<td>97.3%</td>
<td>95.7%</td>
<td>96.2%</td>
<td>93.0%</td>
</tr>
<tr>
<td>AP@.55</td>
<td>95.0%</td>
<td>96.0%</td>
<td>94.6%</td>
<td>95.3%</td>
<td>92.0%</td>
</tr>
<tr>
<td>AP@.60</td>
<td>89.0%</td>
<td>93.8%</td>
<td>91.5%</td>
<td>92.0%</td>
<td>87.2%</td>
</tr>
<tr>
<td>AP@.65</td>
<td>83.7%</td>
<td>84.5%</td>
<td>87.5%</td>
<td>84.4%</td>
<td>82.2%</td>
</tr>
<tr>
<td>AP@.70</td>
<td>78.4%</td>
<td>81.0%</td>
<td>80.0%</td>
<td>77.2%</td>
<td>78.8%</td>
</tr>
<tr>
<td>AP@.75</td>
<td>71.5%</td>
<td>67.3%</td>
<td>72.5%</td>
<td>73.3%</td>
<td>68.6%</td>
</tr>
<tr>
<td>AP@.80</td>
<td>51.6%</td>
<td>51.9%</td>
<td>61.5%</td>
<td>56.7%</td>
<td>54.8%</td>
</tr>
<tr>
<td>AP@.85</td>
<td>28.3%</td>
<td>30.4%</td>
<td>39.2%</td>
<td>29.2%</td>
<td>35.5%</td>
</tr>
<tr>
<td>AP@.90</td>
<td>11.1%</td>
<td>11.2%</td>
<td>15.1%</td>
<td>11.7%</td>
<td>13.7%</td>
</tr>
<tr>
<td>AP@.95</td>
<td>0.9%</td>
<td>1.5%</td>
<td>0.7%</td>
<td>1.3%</td>
<td>1.0%</td>
</tr>
</tbody>
</table>

**TABLE V**: Accuracy w.r.t different values of α in Eq. (17). We test α ∈ {0.45, 0.5, 0.55, 0.6, 0.7}. The accuracy peaks at α = 0.55, which has the corresponding noise level of 0.1. The accuracy then decreases as α becomes larger. The best average performance of our CC-RCNN model is achieved when α = 0.55.
E. Test: Robustness with Respect to Training Data Size

Although CNN-based models have superb performance in many applications, they may require a large amount of training data to achieve low generalization errors. Insufficient training data may lead to overfitting since the number of parameters in CNN is remarkably larger than that of other models. Moreover, it is demanding for human annotators to amplify the training data. Thus, we conduct another sets of experiments to test the robustness of our model with respect to different sizes of the training data.

We trained our CC-RCNN model on eight sizes of training data. We split \{0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8\} of total samples as the training set, the validation and testing sets evenly split the rest of samples. The results of robustness test are shown in Fig. 7. We observe that the accuracy benefits from the larger training set in most cases. However, even if we use 450 events out of 1000 to train our model, we still obtain a good accuracy—AP 44.9%. These results suggest that our DeepDetect yields good accuracy without an enormous amount of training data. It has a notable regularization effect achieved by the reuse of features, and they take full advantage of skip connections to make the model robust.

F. Test: Ablation Study

We conduct ablation experiments to verify the effect of the atrous convolution blocks and the multi-scale architecture. All ablation experiments are conducted with $\alpha = 0.5$ and $\lambda = 10$.

1) Contextual vs Non-Contextual: To demonstrate the importance of using atrous convolutions to incorporate contextual information, we build a non-contextual model (C-RCNN) by directly adding detection branches on top of $D_3 - D_9$. The improvement of performance is indicated by $AP@[.50, .95]$, where the contextual model outperforms the non-contextual counterpart by 13.9 points. More concrete examples are illustrated in Fig. 8. Event #41 in Fig. 8a is the case discussed in Section IV-C. C-RCNN detects event #41 as two individual events due to the second peak in the pattern. In contrast, our contextual model, CC-RCNN, is able of capturing the whole event. Moreover, event #93 in Fig. 8b is missed by C-RCNN. Actually, the classifier gives positive predictions for proposals of event #93. However, the detection of #93 is suppressed by that of event #94 because the predicted beginning of event #94 is inaccurate, which makes the IoU of these two detections above the suppression threshold. Thus, the incorporation of contextual information for individual proposals not only reduces false detections, but also increase the localization accuracy.

2) Multi-scale Architecture: To demonstrate the effect of the proposed multi-scale architecture, which is designed for capturing various lengths of events, we compare the performance of using $D_3 - D_9$ (proposed), $D_4 - D_8$, $D_5 - D_7$ and $D_6$ to make predictions. The results are shown in Table VII. The single-scale model (using $D_6$ only) achieves AP 47.3%, and is outperformed by the proposed multi-scale model by 14.2 points. Figure 9 shows the detection results of two segments. Event #69 and #70 in Fig. 9a span about 4,000 timestamps. Event #73 in Fig. 9b spans about 3,000 timestamps. They are all poorly captured by the single-scale model since their lengths are outside the range that the anchor covers. Though, it is worthwhile to mention that the performance of the single-scale model is still significantly better than the baseline–template matching model. Since we assign positive labels to those anchors having IoU greater than 0.5 with at
TABLE VII: Performance of using different detection branches. The single-scale model has the lowest accuracy. The accuracy grows as more events are covered by multi-scale anchors.

<table>
<thead>
<tr>
<th>ID</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>1102</td>
<td>1698</td>
<td>1236</td>
<td>1526</td>
<td>502</td>
<td>3058</td>
<td>2292</td>
<td>1436</td>
</tr>
</tbody>
</table>

TABLE VIII: The length (in timestamp) of each event in Fig. 10a.

G. Test: Curated and Randomly Selected Detections

To further illustrate the performance of our CC-RCNN model, we present example detections of 5 long segments in Fig. 11 and Fig. 12. All detections are plotted on the top of the ground-truth. Fig. 11 shows the detection results of two curated time series segments, which are selected because of the impressive detection results. The detections of three randomly selected segments are presented in Fig. 12. Although they are not curated examples, we found the detection results still promising.

VII. CONCLUSION

Accurate event detection from 1D times series seismic data is not only important, but also challenging. In this paper, we develop a novel event-wise detection method, DeepDetect, for 1D time series signals. Specifically, a cascaded architecture is developed to generate multi-scale proposals to detect events with various lengths. To take into account the temporal correlation of time series data, we use atrous convolutions with different dilation rates to enrich features of individual proposals. To help with the optimization, we share parameters for branches built on top of multi-scale proposals. For event detection tasks in 1D times series signals, our DeepDetect method is state-of-the-art. In our experimental tests, we compare our new cascaded-contextual region-based convolutional neural network model with a standard method (template matching). Our detection accuracy is significantly higher than that obtained using template matching, especially when the sizes of events are greatly different from one another. We also demonstrate the robustness of our new model to different sizes of the training dataset. To conclude, our new detection model, DeepDetect, yields high performance for a laboratory seismic dataset, therefore it has great potential for event detection in various seismic applications.
Fig. 10: Probability distributions from all detection branches. The input signals and the ground-truth are illustrated in (a). (b)-(h) exhibit probabilities of being an event for each location of the input signal, output by $D_3 - D_9$. Signals are denoted in blue light. Probabilities are denoted as solid lines of varying colors and the dotted red lines indicate the detection threshold of 0.5.

ACKNOWLEDGMENT

This work was co-funded by the Center for Space and Earth Science (CSES) at Los Alamos National Laboratory (LANL), the U.S. DOE Office of Fossil Energy through its Carbon Storage Program, and Institutional Support (LDRD) at Los Alamos.

REFERENCES

[6] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, “Unsuper-
Fig. 11: Curated examples. The two segments are selected because we found the detection results are impressive.


Fig. 12: Randomly selected detection results. Our algorithm yields promising detections for most of the events, although there are some false-positive events obtained (e.g., one at Timestamp of 160,000 of Fig. 12(a)).


DeepDetect: A Cascaded Region-based Densely Connected Network for Seismic Event Detection—A Seismic Application

Yue Wu¹, Youzuo Lin¹,*️, Zheng Zhou², David Chas Bolton³, Ji Liu², and Paul Johnson¹,

Abstract—Automatic event detection from time series signals has wide broad applications. Traditional detection methods detect events primarily by the use of similarity and correlation in data. Those methods can be inefficient and yield low accuracy. In recent years, machine learning techniques have revolutionized many sciences and engineering domains. In particular, the performance of object detection in 2D image data has been significantly improved due to the deep neural network deep neural networks. In this study, we apply—a deep-learning-based method to the detection of events from time series detection method, called “DeepDetect”, to the detect events from seismic signals. However, we find that the direct adaptation of the similar ideas from 2D object detection to our problem faces two challenges. The first challenge is that the duration of earthquake event varies significantly; The other is that the proposals generated are temporally correlated. To address these challenges, we propose a novel cascaded region-based convolutional neural network to capture earthquake events in different sizes—while incorporating contextual information to enrich features for each individual proposal. For proposal. To achieve a better generalization performance, we use densely connected blocks as the backbone of our network. Because some positive events are not correctly annotated, we further formulate the detection problem as a learning-from-noise problem. To verify the performance of our detection methods, we employ our methods to seismic data generated from a bi-axial “earthquake machine” located at Rock the Pennsylvania State University Rock and Sediment Mechanics Laboratory, Penn State University, and we acquire labels with the help of experts. Through our numerical tests, we we—show that our novel detection techniques yield high accuracy. Therefore, our novel deep-learning-based detection methods can potentially be powerful tools for locating identifying events from time series data in various applications.

Index Terms—Convolutional neural network (CNN), seismic signals, time series segmentation, event detection.

I. INTRODUCTION

TIME series data can be acquired through sensor-based monitoring. In the past few years, there have been increased interests in an increased interest to detect useful events out of various from time series datasets for different applications. Among all these problems, seismic monitoring to detect the Earthquake earthquakes has attracted many interests [1, 2]. In this study, we develop a novel event detection method and further employ our method to seismic time series datasets.

Machine learning methods have been successful in object detection to identify patterns. There have been many existing machine learning methods to detect events out of from time series datasets in—for various applications such as epileptic seizure detection from EEG signals and change detection from remotely sensed imagery datasets. Depending on the availability of labeled datasets, all these event detection methods for time series data sets datasets can be categorized into supervised [3, 4, 5, 2, 3, 4, 5] and unsupervised methods [2, 6, 7]. Our study belongs to the supervised category, since we acquire labels for training and evaluation with the help of experts. As for those supervised methods, they are all point-wise point-wise detection methods meaning they classify data point-points at each time stamp. Point-wise Point-wise detection methods can be limited in their detection performance. In particular, those methods can neither accurately localize events nor obtain the number of events. In this study, inspired by the object detection in 2D imagery, we develop a novel event wise detection method event-wise detection method, called “DeepDetect”, to capture each complete event. In other words, our detection methods capture DeepDetect captures the beginning and end coordinates to localize each event from the time series datasets.

Convolutional neural networks (CNN) has achieved promising results in computer vision, image analysis, and many other domains due to the significantly improved computational power ([8, 9, 10, 11]). The State-of-the-Art CNN-based object detection models for 2D imagery mainly consists primarily consist of two steps [12, 13, 14]: a step to generate the region proposals and a step to identify and localize the events within proposals. Specifically, segments of the input data that may include targeting patterns are first used to generate region proposals. A classifier is then employed on each proposal to detect targeting patterns, and a regressor is utilized to localize events within positive proposals. The original proposal generation method for CNN-based detection models is developed in two region-based CNN models, known as R-CNN [14] and Fast R-CNN [13], where fixed methods are used to obtain proposals. Faster-RCNN [12] improves previous models by building region proposal networks (RPN) on top of the final feature map of CNN backbone. Compared with Girshick [13] and Girshick et al. [14], the Faster-RCNN eliminates the additional time spent on proposal generation. To determine whether a proposal is positive or negative, Ren et al.

1: the Earth and Environmental Sciences, Los Alamos National Laboratory, Los Alamos, NM, 87544 USA.
2: the Department of Computer Science, University of Rochester, Rochester, NY 14627.
3: the Department of Geosciences, Penn State University, University Park, PA 16802.
* Correspondence to: Y. Lin, ylin@lanl.gov.
Manuscript received Nov, 2017.
[12] introduces the concept of anchor to denote the region on the input data that a proposal covers. A proposal is considered positive if its corresponding anchor overlaps the ground-truth above a threshold.

In this study, we develop a novel deep neural network detection method for time series datasets. Similar to previous 2D detection models, our model also consists of two steps: proposal generation and event localization. However, a direct adaptation of 2D methods to generate the region proposals does not work well with our 1D seismic time series datasets because the duration of seismic events varies significantly. Therefore, we develop a novel region proposal method to address this issue. In particular, we develop a cascaded network that generates proposals at different levels by including more downsampling layers than regular networks do. Theoretically, events of small size can be captured at shallow layers. As the network becomes deeper, events of large size can be captured due to the increasing size of the receptive field. We add detection branches on feature maps at different depths so that events in various scales can be captured.

Features are critical to the performance of our detection model. Since the classifier and regressor in the second step share the same feature vector obtained from the CNN, enriching features for proposals will boost the detection rate and localization accuracy. Another novelty of our work is the incorporation of contextual information for each individual proposal. Although the importance of contextual information has been emphasized for imagery segmentation [15, 16, 17], there are surprisingly few detection models taking into account contextual information on the proposal level. As for the time series seismic signals, proposals are temporally correlated. Utilizing each proposal individually generates many false-positive detections. This is because these proposals may be part of some large events, and our detection method should be able to distinguish those small signal segments from large events. Considering this, we enrich features of each proposal by incorporating contextual information.

Because of the cascaded structure, the number of parameters in our model may significantly increase. To obtain a better generalization performance, we build our model—DeepDetect based on densely connected network networks (DenseNet) [8]. The core idea of DenseNet is to reuse features learned from shallow layers, which enables us to maintain a reasonable number of parameters even if the network becomes substantially deep. Another strategy we use to address overfitting is to share the parameters of the sibling detector and regressor. This is reasonable since we are interested in capturing specific patterns regardless of their size.

Another challenge of our event detection problem is that due to the variation in the event patterns and density, it is impractical for domain experts to accurately annotate all events because the pattern of seismic events is not as obvious as the one in image objects. Those omitted events may bias the classifier for proposals. To alleviate the impact of mis-labeled positive events, we further formulate the proposal classification as a learning-from-noise problem. Inspired by Natarajan et al. [18], we use a label-dependent loss function for the classifier.

We test our detection models on seismic time series data and compare the experimental results obtained using the proposed cascaded contextual region-based CNN (CC-RCNN) and the traditional template matching method. We use AP@[0.5, 0.95] as the evaluation metric. Average precision (AP) calculates the averaged maximum precision at each recall value. We calculate 10 APs with the intersection over union (IoU) of [0.5:0.95:0.05] as the metric to identify true positive detections and then take the average. We also conduct ablation experiments to verify the effect of the multi-scale architecture and the incorporation of contextual information. The experiment results demonstrate that our deep-learning-based model significantly outperforms the template matching method and the ablation studies further show that the incorporation of contextual information for each individual proposal not only reduces false-positive detections, but also significantly increases the event localization accuracy. Also the utilization of label-dependent loss further boosts the performance of our detection models. To summarize, our contributions can be listed as follows:

- Extend region-based convolutional neural networks to time series scenarios;
- Propose a cascaded structure to generate multi-scale proposals to efficiently capture events in varying lengths;
- Incorporate contextual information for each proposal to further boost the detection accuracy;
- Conduct experiments on seismic time series data and obtain promising results—achieving average precision (AP)@[0.5, 0.95] of 63.8%, which significantly outperforms the template matching method.

The rest of the paper is organized as follows. Section II briefly reviews related works on event detection and object detection. Section III gives background knowledge about our method. The proposed method is elaborated in Section IV. Implementation details of the proposed methods and counterpart methods are described in Section V. Experiment results are provided in Section VI. Section VII concludes this paper.

II. RELATED WORK

Our study is related to both event detection for 1D time series datasets and object detection for 2D imagery datasets.

1) Detection Methods for 1D Datasets

Time Series Data: There are many event detection methods in various applications. In seismology, STA/LTA is the most popular used detection method due to its simplicity [19, 20]. STA/LTA computes the ratio of short-term average energy and long-term average energy on multiple receivers. If a seismic event is detected at a minimum of four stations, it is considered as an event. The ratio exceeds a pre-defined threshold, a detection is declared. However, STA/LTA fails to detect earthquakes or yields many false detections in challenging situations such as signal with low signal-to-noise ratio, or overlapping events. Compared to STA/LTA, Autocorrelation autocorrelation yields a much higher detection rate [21]. Autocorrelation is an exhaustive “many-to-many” detection method. It searches for similar waveforms when the desired signal waveform is unknown. The major disadvantage with autocorrelation is its lies in its notoriously expensive
computational cost—its computational complexity—which scales quadratically with data duration. Template matching is a detection method that yields a good balance between accuracy and computational complexity [22, 23]. Template matching is a “one-to-many” detection method. It computes the correlation coefficient of a template waveform with the candidate waveform data. A detection is claimed when the correlation coefficient value is above the user-defined threshold value. Template matching has been proven to be efficient and successful for different seismic applications: microseismic monitoring in geothermal fields [24], oil and gas reservoirs [25], nuclear monitoring [26], and tectonic tremor [23]. However, template matching usually performs poorly when events are dissimilar. Yoon et al. [2] recently developed an unsupervised-event detection approach called the fingerprint and similarity thresholding (FAST) method and applied it to detect earthquake-earthquakes out of seismic datasets. FAST creates “fingerprints” of waveforms by extracting key discriminative features—then group and then groups similar fingerprints together within a database to facilitate the fast and scalable search for similar fingerprint pairs.

There are many other event detection methods developed in other application domains. Oehmcke et al. [3] employed local outlier factor to detect events from marine time series data. To further improve results, dimensionality reduction methods are employed by the authors to the datasets. In the work of Batal et al. [5], an event detection method was developed based on recent temporal patterns. The detection algorithm mines time-interval patterns backward in time, starting from patterns related to the most recent observation. The authors further applied their detection method to health care data of diabetic patients. McKenna et al. [27] developed a binomial event discriminator (BED) method. BED uses a failure model based on the binomial distribution to determine the probability of an event within a time segment. They applied the method to hydrological datasets to detect events. The effectiveness of their model is demonstrated with hydrological datasets.

2) CNN-based Detection Methods for 2D Data: Image Data: Ren et al. [12] developed the faster RCNN method, of which a window is slid on the final feature map of the fourth stage of ResNet [9] to generate proposals. The authors use nine different anchors with three various sizes (128, 256, 512) and three ratios of height/width (1:1, 1:2, 2:1) to determine regions that a proposal covers. To make anchors more accurate, Cai et al. [28] developed the multi-scale CNN methods, consisting of a proposal sub-network to generate multi-scale proposals at three stages of VGG [10] network. The authors then built detectors on top of each proposal branch.

III. BACKGROUND AND RELATED WORK

A. Template Matching

Template matching (TM) [29] is widely used in the seismology community. It calculates the similarity of a template with successive windows from continuous waveform data. The commonly used similarity metric is normalized cross-correlation (CC),

\[
CC(a, b) = \frac{\langle a, b \rangle}{\|a\|_2 \|b\|_2} = \frac{\sum_{i} a_i b_i}{\sqrt{\sum_{i} a_i^2} \sqrt{\sum_{i} b_i^2}},
\]

where \( a, b \) are vectorized time series signals. The detection threshold of \( \tau \) is given as

\[
\tau = \mu \cdot \text{median absolute deviation (MAD)},
\]

where \( \mu \) is usually chosen as 9 [30, 31, 32]. For a univariate set \( X_1, X_2, ..., X_n \), MAD can be calculated as

\[
\text{MAD}(X) = \text{median}(|X_i - \text{median}(X)|).
\]

B. Densely Connected Network

Densely connected network (DenseNet) [8] is an improved version of the residual network [9]. Both ResNet and DenseNet are discussed in this section.

1) ResNet Block: The major breakthrough of ResNet is the application of skip connections between residual blocks, which can be denoted as

\[
x_{i+1} = x_i + W' \ast (\sigma(R(W \ast (\sigma(B(x_i)))))
\]

where \( x_i \) is the input feature to the \( i \)-th residual block, \( W' \) and \( W' \) are weight matrices, the operator of \( \ast \) denotes convolution, \( B \) denotes batch normalization (BN) [33], \( \sigma(x) = \max(0, x) \) [34]. Eq. (4) forms a building block in ResNet. Skip connections are implemented by summing up the input of the block and the output of a set of convolution layers. The existence of skip connections weakens the importance of each individual path, so that the model behaves like an ensemble of small networks. Although ResNet immediately topped most of benchmarks, a few drawbacks still need to be addressed.

1) The number of parameters becomes extremely large with hundreds layers of convolutions. 2) The network may not benefit from “going deep” due to the gradient vanishing problem, as indicated in Veit et al. [35].

2) DenseNet Block: A DenseNet block is then formulated as

\[
x_{i+1} = H([x_0, x_1, ..., x_i])
\]

\[
H(x) = W \ast (\sigma(B(x))),
\]

where \([x_0, x_1, ..., x_i] \) denote the input of each convolution layer in the block, the bracket “( )” denotes the concatenation of all outputs of previous layers. A DenseNet block is illustrated in Fig. 1.

Thus, the output of a layer in one DenseNet block is densely connected with outputs of all deeper layers in the same block by means of concatenation. Thus, it fully exploits the advantage of skip connections. Moreover, features from...
shallow layers are reused by deep layers, which reduces the number of parameters, and the gradient vanishing problem is further alleviated by the concatenation layers.

The feature dimension \( d_{l+1} \) of \( x_{l+1} \) is calculated as

\[
d_{l+1} = d_0 + k \cdot l,
\]

where \( k \) (a.k.a the growth rate) is the number of filters used for each convolution layer.

C. Atrous Convolution

Atrous convolution convolves input nodes with a dilation rate \( d \), denoting the stride for each convolved location on input nodes. The output node \( y_i \) of an atrous convolution layer is calculated as

\[
y_i = \sum_{k=1}^{K} x_{i+d \cdot k} \cdot w_k,
\]

where \( k \) is the dimension, \( x_i \) is the input node and \( w \in \mathbb{R}^K \) is the kernel.

The regular convolution can be seen as a special case of atrous convolution with \( d = 1 \). Atrous convolution was first proposed in [36] to address the low-resolution problem caused by downsampling layers (pooling, convolution with stride, etc). Atrous convolution essentially involves distant information by covering larger regions of input signals while maintaining the same number of parameters.

IV. PROPOSED METHODS

A. Network Architecture

As illustrated in Table I, our network DeepDetect is inspired by DenseNet. All convolution kernels in our network are 1D because of the input of 1D time series data. The “Conv7, 64, /2” denotes using \( 64 \times 1 \times 7 \) convolution kernels with stride 2. The same routine applies to max-pool and avg-pool. The brackets denote DenseNet blocks illustrated in Fig. 1.

Specifically, we use “[conv3, 20] × 6” to denote a DenseNet block where 6 convolution layers are applied, each followed by batch normalization and ReLU activation layers. \( D_i \), \( T_i \) denote DenseNet blocks and transition blocks, respectively. All transition blocks have an average pooling layer to downsample the signal by 2, while \( T_3 - T_9 \) have an extra \( 1 \times 1 \) convolution layer to reduce the feature dimension by half. As previously discussed, our model is designed for capturing events with various different durations, so we use the output of each \( D_3 - D_9 \), having strides \( 16, 1024, 16, 32, 64, 128, 256, 512, 1024 \) on the input signals, as proposals, then detection branches (the classification and regression layer) are built on the top of these multi-scale proposals. Since small events greatly outnumber large events, we share the detection branches for all scales to make our model robust. To achieve this, we set growth rate \( k = 12 \) for \( D_1, D_2 \), \( k = 20 \) for \( D_3 - D_9 \), so that all proposals have the same feature dimension 240.

### Table I: DenseNet: The cascaded network architecture of DeepDetect for event detection. Conv7, 64, /2 denotes using \( 64 \times 1 \times 7 \) convolution kernels with stride 2. The same routine applies to max-pool and avg-pool. The brackets denote DenseNet blocks illustrated in Fig. 1.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Layers</th>
<th>Dim.</th>
<th>Anchor Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>conv7, 24, /2</td>
<td>L/2 × 24</td>
<td>-</td>
</tr>
<tr>
<td>Pool</td>
<td>max-pool3, /2</td>
<td>L × 24</td>
<td>-</td>
</tr>
<tr>
<td>( D_1 )</td>
<td>[conv3, 12] × 6</td>
<td>L/4 × 24</td>
<td>-</td>
</tr>
<tr>
<td>( T_1 )</td>
<td>avg-pool2, /2</td>
<td>L/8 × 96</td>
<td>-</td>
</tr>
<tr>
<td>( D_2 )</td>
<td>[conv3, 12] × 6</td>
<td>L/8 × 168</td>
<td>-</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>avg-pool2, /2</td>
<td>L/16 × 168</td>
<td>-</td>
</tr>
<tr>
<td>( D_3 )</td>
<td>[conv3, 12] × 6</td>
<td>L/16 × 240</td>
<td>128</td>
</tr>
<tr>
<td>( T_3 )</td>
<td>conv1, 120</td>
<td>L/32 × 120</td>
<td>-</td>
</tr>
<tr>
<td>( D_4 )</td>
<td>[conv3, 20] × 6</td>
<td>L/32 × 240</td>
<td>256</td>
</tr>
<tr>
<td>( T_4 )</td>
<td>conv1, 120</td>
<td>L/64 × 120</td>
<td>-</td>
</tr>
<tr>
<td>( D_5 )</td>
<td>[conv3, 20] × 6</td>
<td>L/64 × 240</td>
<td>512</td>
</tr>
<tr>
<td>( T_5 )</td>
<td>conv1, 120</td>
<td>L/128 × 120</td>
<td>-</td>
</tr>
<tr>
<td>( D_6 )</td>
<td>[conv3, 20] × 6</td>
<td>L/128 × 240</td>
<td>1024</td>
</tr>
<tr>
<td>( T_6 )</td>
<td>conv1, 120</td>
<td>L/256 × 120</td>
<td>-</td>
</tr>
<tr>
<td>( D_7 )</td>
<td>[conv3, 20] × 6</td>
<td>L/256 × 240</td>
<td>2048</td>
</tr>
<tr>
<td>( T_7 )</td>
<td>conv1, 120</td>
<td>L/512 × 120</td>
<td>-</td>
</tr>
<tr>
<td>( D_8 )</td>
<td>[conv3, 20] × 6</td>
<td>L/512 × 240</td>
<td>4096</td>
</tr>
<tr>
<td>( T_8 )</td>
<td>conv1, 120</td>
<td>L/1024 × 120</td>
<td>-</td>
</tr>
<tr>
<td>( D_9 )</td>
<td>[conv3, 20] × 6</td>
<td>L/1024 × 240</td>
<td>8192</td>
</tr>
</tbody>
</table>

B. Anchors

Anchor is the effective region of the input signals that a proposal is responsible for. In most cases, it is used to decide the label for that proposal. In our time series data, an anchor indicates two coordinates representing the beginning and the end of each proposal. We assign the anchor size 128 to proposals in \( D_3 \), and it doubles for the next scale. Proposals in \( D_9 \) have the largest anchor size 8192. These settings are determined by the length distribution of events in our data. It is worthwhile to mention that the amount of shift-shift between adjacent proposals is determined by the stride of that stage on the input signals. For example, the shift between two adjacent anchors of \( D_3 \) is 16 timestamps. Specifically, the stride is 1/8 of the anchor size for each \( D_3-D_9 \).

C. Proposals with Contextual Information

Features are critical to detection. In time series data, it is important to take into consideration the temporal correlations among neighboring proposals. Considering Only considering
features from each individual proposal only will result in many false detections.

Figure 2a illustrates a perfect individual event in time series data. The signal amplitude keeps at the consistent—remains at a constant level before a major event comes. As the event vanishes, the signal amplitude decreases to the previous level returns to the background. However, it can be possible that the signal amplitude does not decrease monotonically or the major event may last longer than usual. Both cases will lead to the scenario when truncations from a major event are mis-detected as several small events. Figure 2b illustrates an example of a false detection. The whole segment of signals in Fig. 2b (denoted as “top”) is a single event. However, if we only focus on a truncation of that, i.e., the “bottom” part as shown in Fig. 2b, we may mistakenly consider this truncation as an individual event. Therefore, in order to detect each event as a whole, it is necessary to check the preceding and succeeding patterns for each proposal.

We build atrous convolution blocks on seven proposal layers, $D_{3\times3} - D_{9}$. The atrous convolution block is illustrated in Fig. 3. The dilation rate in atrous convolution indicates the number of skipped proposals at each convolved location. We set dilation rates to be 4, 8, and 12 for proposals in all scales. These dilation rates are inspired by the amount of shifts of adjacent proposals. Anchors of adjacent proposals shift only a little, hence the features of adjacent proposals tend to be similar. In contrast, atrous convolution is capable of incorporating contextual information. For the other extreme, the shifts of anchors should not be too large since the information from far away will be irrelevant to the target proposal. With dilation rates of 4, 8, 12, the shifts between the target proposal and contextual proposals are 0.5, 1, 1.5 of the anchor size. The blocks in red, green, and blue shown in Fig. 3 are the outputs of atrous convolutions with 4, 8, and 12 as dilation rates, followed by batch normalization and activation layers. All convolutions are $1 \times 3$, with 240 kernels. We generate new proposals with contextual information by concatenating outputs using three dilation rates and the target proposal. The new proposal includes four times as many features as the target proposal. In order to keep the number of features unchanged, we further employ a $1 \times 1$ convolution layer. To summarize, we employed atrous convolution with 3 dilation rates on proposals layers so that the features of each individual proposal are enriched by proceeding—preceding and succeeding proposals, while maintaining its own features.

Fig. 2: (a) top: A perfect individual event in time series data. (a) bottom: A zoomed-in truncation of (a) top. (b) top: A segment of time series signals, and the whole segment is labeled as an event. (b) bottom: A truncation of (b) top. The green and red boxes indicate the beginning and end of an event, respectively.

D. Sibling Branches for Detection and Localization

Fig. 3: The atrous convolution block. We aim at enriching features of individual proposals by convolving with proceeding—preceding and succeeding proposals. In this figure, the target proposal is at the center. Atrous convolutions with different strides are applied to capture contextual information from nearby to further proposals.

Fig. 4: An illustration of multi-scale detections. There are three—three proposals (green nodes) should be—are considered positive since they have the intersection over union (IoU) with the ground-truth above the threshold. Ground-truth are indicated at the bottom with green and red dots, which denote beginnings and ends, respectively.

We add a classification branch and a regression branch on each proposal. The classification branch is first used to detect whether a proposal includes an event or not. For each positive
proposal, we further apply a regressor to localize the event within. We use a joint loss function to optimize classification and regression branches simultaneously.

We assign a positive label to a proposal if its anchor has the ratio of intersection over union (IoU) above 0.5 with at least one ground-truth event. Proposals are assigned a negative label if the highest IoU of their anchors with the ground-truth is below 0.3. Neutral proposals (IoU ∈ [0.3, 0.5]) do not contribute to the loss. To localize the event within a proposal, two offsets: $d_x$ and $d_w$ are captured to transform the anchor to real coordinates by

$$G^*_x = P_x d_x + P_x,$$
$$G^*_w = P_w \exp(d_w),$$
(9)
$$G_x = P_x d_x + P_x,$$
$$G_w = P_w \exp(d_w),$$
(10)

where $P_x, P_w$ are the center and length of an anchor, $G^*_x, G^*_w, G_x, G_w$ are the center and length of the prediction. Figure 4 gives an illustration of this process, where three nodes are positive so they also have a localization loss.

Another challenge in our time series data is that not all events in the training set are annotated, which is caused by the fact that some patterns are difficult for our annotators to decide. Due to this problem, negative labels are noisy in our task. To address this issue, we employ label-dependent cost function for the classifier.

The label-dependent cost function was initially proposed in a couple of works [37, 38], which is known as weighted logistic regression and biased support vector machine, respectively. The core idea of “label-dependent” is to apply separate loss functions for positive and negative sets

$$J(g(x)) = \frac{1}{X} \left( \alpha \sum_{x \in X_+} l(g(x)) + \beta \sum_{x \in X_-} l(g(x)) \right),$$
(11)

where $l$ can be any 0-1 loss functions, $X_+, X_-$ denote the observed positive and negative sets, and $g$ is a linear score function.

To obtain the optimal weight parameters $\alpha^*$ and $\beta^*$, [18] set $\rho_{+1} = P(\hat{Y} = 1|Y = 1)$ and $\rho_{-1} = P(\hat{Y} = 1|Y = -1)$ to calculate

$$\alpha^* = \frac{1 - \rho_{+1} + \rho_{-1}}{2},$$
$$\beta^* = 1 - \alpha^*$$

where $\hat{Y}$ denotes the observed label, $Y$ denotes the true label and the conditional probability $P$ denotes the probability of a sample being wrongly labeled. It can be shown that by employing the optimal parameters of $\alpha^*$ and $\beta^*$, the resulting classifier can make predictions of $\sigma \left( \frac{\text{sign}(g(x) - 1/2)}{\sqrt{\sigma}} \right)$ with noisy data [18]. We use the similar parameter estimation approach to our datasets by setting $\rho_{+1} = 0$, since the noise only exists in negative samples.

1) Loss Function: We develop a joint loss function $L$ including a classification cost function $L_{cls}$ and a regression cost function $L_{regr}$

$$L(d_{cls}, d_x, d_w, t_{cls}, t_x, t_w) = L_{cls}(d_{cls}, t_{cls}) + \lambda 1\{t_{cls} = 1\} \sum_{u \in \{x, w\}} L_{regr}(d_u, t_u),$$
(16)

where $1\{t_{cls} = 1\}$ is the indicator function indicating only positive proposals contribute to the regression loss, and $d_{cls}^j, d_x^j, d_w^j$ are the predictions of the $i^{th}$ proposal's class score, center and length offsets, respectively, and $t_{cls}^j, t_x^j, t_w^j$ are the corresponding ground-truth of the $i^{th}$ proposal's class score, center and length offsets, respectively, and $\lambda$ is the regularization parameter controlling the balance between $L_{cls}$ and $L_{regr}$.

The classification cost function $L_{cls}$ is defined as a label-dependent logistic loss

$$L_{cls}(d_{cls}, t_{cls}) = \alpha 1\{t_{cls} = 1\} \log(1 + e^{-d_{cls}^j}) + (1 - \alpha) 1\{t_{cls} = -1\} \log(1 + e^{d_{cls}^j}),$$
(17)

where $\alpha$ is the hyper-parameter.

The regression cost function $L_{regr}$ is defined as the smoothed $L_1$ loss as proposed in [13]:

$$L_{regr}(d_u, t_u) = \text{smooth}_{L_1}(t_u - d_u),$$
(18)

where $d_u, t_u$ denote the predicted and the target values, and smooth_{L_1}(x) is

$$\begin{cases}
\frac{0.5x^2}{|x|} & \text{if } |x| < 1 \\
0.5 & \text{otherwise}
\end{cases}$$

According to Eqs. (9) and (10), $t_x$ and $t_w$ can be obtained by

$$t_x = \frac{G_x - P_x}{P_w},$$
$$t_w = \ln(G_w/P_w),$$

$$t_x = \frac{G_x^* - P_x}{P_w},$$
$$t_w = \ln(G_w^*/P_w),$$

where $G_x$ and $G_w$ are the ground-truth of the center and length of the event.

2) Share Weights for Robustness: To capture events with dramatically varying durations, we make multi-scale predictions on output layers with different sizes of receptive fields. However, events with different lengths are not equally distributed. In other words, small events greatly outnumber large events. Our model should capture patterns from all events regardless of their durations or magnitudes, amplitudes; hence
we share the weights of contextual atrous convolution layers, sibling classification and regression branches built on top of $P_3^2 D_3^2 - P_5 D_3$. Weight sharing makes our model robust and help with the optimization because predictions in all scales equally contribute to the loss function.

V. IMPLEMENTATION DETAILS

A. Template Matching

We use events in the training set as templates. For each template, CC is calculated at each sliding location of the time series data. We set the detection threshold $\mu = 8$ in Eq. (2), which is determined by the validation set. For multi-detections of a single event, the detection with the highest CC is kept and all other detections are discarded. The beginning and the end of each detection are determined by those of the template.

B. Proposed Model

1) Optimization: The proposed model has approximately 3 million parameters. For each mini-batch iteration, we feed a 24,576-timestamp (6 ms) time series segment with 0.5 overlapping rate so that if the end point of an event lies outside of the segment, that event will be roughly at the center of the next segment. In our dataset, large events are rare comparing to small events. To account for this characteristic, we generate less proposals from large events than those from small events for training. In particular, Table II shows how we select the number of proposals from each detection branch.

<table>
<thead>
<tr>
<th>stage</th>
<th>$C_3$</th>
<th>$C_4$</th>
<th>$C_5$</th>
<th>$C_6$</th>
<th>$C_7$</th>
<th>$C_8$</th>
<th>$C_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td># of proposals</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>32</td>
<td>32</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II: Number of proposals selected from each scale.

To further simplify the optimization, we also make sure the ratio of positive and negative proposals is 1:1. If positive or negative proposals are insufficient, we use neutral ones as negative proposals. Adam optimizer [39] is applied with the initial learning rate of 5e-4. The learning rate is multiplied by 0.1 for every ten epochs. Each mini-batch data is subtracted by the mean and divided by the standard deviation before feeding into the network. The implementation is built on Our model is implemented with TensorFlow [40].

2) Inference: Same as Similarity as for the training process, we feed 24,576 timestamp segments with the overlapping rate of 0.5 each time window. Predictions are first generated on all proposals, and then proposals, then we apply non-maximum suppression (NMS) to reduce multi-detections for a single event. Since events in time series data are rarely overlapped, we set the IoU threshold of NMS to be 0.05.

VI. EXPERIMENT

A. Data

We use time series data acquired at the Rock and Sediment Mechanics Laboratory of Penn State University. The dataset is a time-amplitude representation generated by a. The experiment was conducted using the double-direct shear apparatus to mimic real Earthquake [11-12]. There are shear configuration where two laboratory fault zones are sandwiched between two forcing blocks and sheared simultaneously.

Acoustical data is recorded from two broadband (.02-2MHz) piezoelectric transducers with a central frequency of 500kHz. The transducers are embedded within a 10x10 (cm$^2$) steel block and placed adjacent to the fault zone [43]. Acoustical data is sampled continuously at 4MHz with a 14-bit Verasonics data acquisition system. We hand-picked 1000 acoustic emissions (AEs) from the raw time series data between 2368.9-2369.7 (s). During this section of the experiment, the normal stress is at 3.157,566 MPa. Furthermore, this time corresponds to the first two seconds of the stick-slip cycle. For a single AE, we picked the beginning and end of the event based on the amplitude of the signal, duration of the signal and the characteristic shape of the signal. 1000 seismic events are manually picked by experts. We use 800 events for training, 100 events for validating, and 100 events for testing.

We calculate the length The length distribution of all events is shown in Fig. 5. The largest event spans more than 7,000 timestamps (1.7 ms) while the smallest event spans few hundreds. The mean and median lengths of events are both about 1,500 timestamps.

B. Metric

We use average precision (AP) to evaluate our models. AP first calculates the precision-recall curve, then averages maximum precisions for each unique recall. For AP@.5, a detection is considered as true positive if it has IoU above 0.5 with a ground-truth event. If there are multiple detections for one event, only one detection is considered true positive, others are considered false positive. In this study, we use AP@[.5, .95], which is used in MS COCO object detection dataset [44]. To obtain AP@[.5, .95], we calculate 10 APs using IoUs from
0.5 to 0.95 with stride 0.05, then take the average of 10 APs. The IoU for two 1D segments $A_0A_1, B_0B_1$ is calculated as:

$$\text{IoU}(A_0A_1, B_0B_1) = \frac{\max(y_a - x_a, 0)}{y_b - x_b}.$$ 

### C. Test: Overall Performance

<table>
<thead>
<tr>
<th></th>
<th>TM</th>
<th>CC-RCNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP@.50</td>
<td>22.0%</td>
<td>95.7%</td>
</tr>
<tr>
<td>AP@.55</td>
<td>12.4%</td>
<td>94.6%</td>
</tr>
<tr>
<td>AP@.60</td>
<td>7.4%</td>
<td>91.5%</td>
</tr>
<tr>
<td>AP@.65</td>
<td>4.6%</td>
<td>87.5%</td>
</tr>
<tr>
<td>AP@.70</td>
<td>3.8%</td>
<td>80.0%</td>
</tr>
<tr>
<td>AP@.75</td>
<td>2.4%</td>
<td>72.5%</td>
</tr>
<tr>
<td>AP@.80</td>
<td>1.1%</td>
<td>61.5%</td>
</tr>
<tr>
<td>AP@.85</td>
<td>0.5%</td>
<td>39.2%</td>
</tr>
<tr>
<td>AP@.90</td>
<td>0.3%</td>
<td>15.1%</td>
</tr>
<tr>
<td>AP@.95</td>
<td>0.1%</td>
<td>0.7%</td>
</tr>
<tr>
<td>AP@[.50, .95]</td>
<td>5.5%</td>
<td>63.8%</td>
</tr>
</tbody>
</table>

TABLE III: Accuracy results obtained using template matching and our CC-RCNN model. We notice that our CC-RCNN model consistently yields better detection accuracy than the template matching method.

We provide detection results in Table III obtained using the template matching (baseline), and our deep-learning-based CC-RCNN. Table III shows that our CC-RCNN model consistently yields better detection accuracy than the template matching method. The average AP value of our method is 63.8% compared to the value of 5.5% obtained by the TM method.

To visualize the detection results using our method and the TM method, we provide the example detections of these two models in Fig. 6. The results obtained using our method are denoted by “×”, and those obtained by template matching are denoted in “•”. The ground-truth (“•”) is also provided. Although TM has the ability to detect some events, it has poor localization performance. For instance, TM only captures the second half part of event #6 in Fig. 6a. Also, TM cannot capture event #7 as a whole since each template works separately. The reason for the poor localization is that the lengths of TM detections are simply determined by those of templates, and most events have lengths that no templates can match. On the other hand, the proposed CNN-based method is capable of accurately detecting, as well as localizing multi-scale events due to the cascaded design. To capture events shown in Fig. 6b is more challenging than those in (a) since some of events in (b) vary dramatically in lengths (such as #57, #58) and yield irregular patterns (such as #60). TM detect #57 - #60 as a whole event, while CC-RCNN is able to detect and localize each individual event accurately.

Based on results shown in Table. III and Fig. 6, our detection methods yield much higher accuracy than template matching method.

### D. Test: Hyper-Parameters

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>AP@[.50, .95]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>48.2%</td>
</tr>
<tr>
<td>1</td>
<td>60.4%</td>
</tr>
<tr>
<td>10</td>
<td>61.5%</td>
</tr>
<tr>
<td>100</td>
<td>56.0%</td>
</tr>
</tbody>
</table>

TABLE IV: Accuracy w.r.t different values of $\lambda$ in Eq. (16). We test $\lambda \in \{0.1, 1, 10, 100\}$. The accuracy peaks when $\lambda = 10$.

Hyper-parameters play an important role in our model to achieve high performance. Specifically, the selection of $\lambda$ value in Eq. (16) and $\alpha$ value in Eq. (15) are critical to the detection accuracy using our label-dependent loss function.

We provide results in Table IV to illustrate the performance of our algorithm using different $\lambda$ values in Eq. (16). Similar
to Ren et al. [12], different values of \( \lambda \in \{0.1, 1, 10, 100\} \) are tested. We observe that the performance of our model can be impacted notably by using different \( \lambda \) values. The best performance is achieved with \( \lambda = 10 \), which is therefore used for all the tests implemented within the work.

To demonstrate the effectiveness of our selected different values of \( \alpha \) value, we further provide detection results using our label-dependent loss function in Eq. (17). The performance on different noise parameters, \( \alpha \in \{0.45, 0.5, 0.55, 0.6, 0.7\} \), is reported in Table V. The best average performance of our CC-RCNN model is achieved when \( \alpha = 0.55 \) with the corresponding noise level of 0.1. This is reasonable since the noise only exists in negative samples, and the noise level is low. The accuracy decreases when either the noise level becomes larger (\( \alpha = 0.55, 0.6, 0.7 \)) or assuming that the noise exists in positive samples (\( \alpha = 0.45 \)).

Through these sets of tests on hyper-parameter, we obtain the best combination to use for our dataset i.e., \( \lambda = 10 \) and \( \alpha = 0.55 \).

### E. Test: Robustness with Respect to Training Data Size

Although CNN-based models have superb performance in many applications, they may require a large amount of training data to achieve low generalization errors. Insufficient training data may lead to overfitting since the number of parameters in CNN is remarkably larger than that of other models. Moreover, it is demanding for human annotators to amplify the training data. Thus, we conduct another sets of experiments to test the robustness of our model with respect to different sizes of the training data.

We trained our CC-RCNN model on eight sizes of training data. We split \( \{0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8\} \) of total samples as the training set, the validation and testing sets evenly split the rest of samples. The results of robustness test are shown in Fig. 7. We observe that the accuracy benefits from the larger training set in most cases. However, even if we use 450 events out of 1000 to train our model, we still obtain a good accuracy–AP 44.9%. These results suggest that our DenseNet-based model DeepDetect yields good accuracy without an enormous amount of training data. It has a notable regularization effect achieved by the reuse of features, and they take the full advantage of skip connections to make the model robust.

#### F. Test: Ablation Study

We conduct ablation experiments to verify the effect of the atrous convolution blocks and the multi-scale architecture. All ablation experiments are conducted with \( \alpha = 0.5 \) and \( \lambda = 10 \).

1) Contextual vs Non-Contextual: To demonstrate the importance of using atrous convolutions to incorporate contextual information, we build a non-contextual model (C-RCNN) by directly adding detection branches on top of \( D_3 \) – \( D_9 \). The improvement of performance is indicated by AP@[.50, .95], where the contextual model outperforms the non-contextual counterpart by 13.9 points. More concrete examples are illustrate in Fig. 8. Event #41 in Fig. 8a is the case discussed in

![Fig. 7: The APs achieved by using different training data sizes.](image)

The AP peaks when using 800 samples for training. Even with 450 samples, which is less than a half, our model still has a good performance.

<table>
<thead>
<tr>
<th>Training Percentage</th>
<th>C-RCNN</th>
<th>CC-RCNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>44.9%</td>
<td>63.8%</td>
</tr>
<tr>
<td>50</td>
<td>50.4%</td>
<td>59.2%</td>
</tr>
<tr>
<td>55</td>
<td>49.8%</td>
<td>57.4%</td>
</tr>
<tr>
<td>60</td>
<td>52.0%</td>
<td>52.3%</td>
</tr>
<tr>
<td>65</td>
<td>52.3%</td>
<td>52.3%</td>
</tr>
<tr>
<td>70</td>
<td>57.4%</td>
<td>57.4%</td>
</tr>
<tr>
<td>75</td>
<td>59.2%</td>
<td>59.2%</td>
</tr>
<tr>
<td>80</td>
<td>63.8%</td>
<td>63.8%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event #41 in Fig. 8a</th>
<th>C-RCNN</th>
<th>CC-RCNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>44.9%</td>
<td>63.8%</td>
</tr>
<tr>
<td>50</td>
<td>50.4%</td>
<td>59.2%</td>
</tr>
<tr>
<td>55</td>
<td>49.8%</td>
<td>57.4%</td>
</tr>
<tr>
<td>60</td>
<td>52.0%</td>
<td>52.3%</td>
</tr>
<tr>
<td>65</td>
<td>52.3%</td>
<td>52.3%</td>
</tr>
<tr>
<td>70</td>
<td>57.4%</td>
<td>57.4%</td>
</tr>
<tr>
<td>75</td>
<td>59.2%</td>
<td>59.2%</td>
</tr>
<tr>
<td>80</td>
<td>63.8%</td>
<td>63.8%</td>
</tr>
</tbody>
</table>

### TABLE V: Accuracy w.r.t different values of \( \alpha \) in Eq. (17).

We test \( \alpha \in \{0.45, 0.5, 0.55, 0.6, 0.7\} \). The accuracy peaks at \( \alpha = 0.55 \), which has the corresponding noise level of 0.1. The accuracy then decreases as \( \alpha \) becomes larger. The best average performance of our CC-RCNN model is achieved when \( \alpha = 0.55 \).

### TABLE VI: This table demonstrates the effect of atrous convolutions for incorporating contextual information.
Section IV-C. C-RCNN detects event #41 as two individual events due to the second peak in the pattern. In contrast, our contextual model, CC-RCNN, is able to capture the whole event. Moreover, event #93 in Fig. 8b is missed by C-RCNN. Actually, the classifier gives positive predictions for proposals of event #93. However, the detection of #93 is suppressed by that of event #94 because the predicted beginning of event #94 is inaccurate, which makes the IoU of these two detections above the suppression threshold. Thus, the incorporation of contextual information for individual proposals not only reduces false detections, but also increase the localization accuracy.

2) Multi-scale Architecture: To demonstrate the effect of the proposed multi-scale architecture, which is designed for capturing various length-lengths of events, we compare the performance of using $D_3 - D_9$ (proposed), $D_4 - D_8$, $D_5 - D_7$, and $D_6$ to make predictions. The results are shown in Table VII. The single-scale model (using $D_6$ only) achieves AP 47.3%, and is outperformed by the proposed multi-scale model by 14.2 points. Figure 9 shows the detection results of two segments. Event #69 and #70 in Fig. 9a span about 4,000 timestamps. Event #73 in Fig. 9b spans about 3,000 timestamps. They are all poorly captured by the single-scale model since their lengths are outside the range that the anchor covers. Though, it is worthwhile to mention that the performance of the single-scale model is still significantly better than the baseline-template matching model. Since we assign positive labels to those anchors having IoU greater than 0.5 with at least one ground-truth event, $D_6$ is theoretically capable of capturing events with 512-2,048 timestamps. The majority of events in our dataset can be captured with this range. To further demonstrate the effectiveness of each branch, we plot their probability distributions in Fig. 10.
have been scaled to [0, 1]. A testing segment is shown in Fig. 10a, with 8 events to be captured. The length of each event is shown in Table VIII. Figure 10b-10h display the

<table>
<thead>
<tr>
<th>ID</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>1102</td>
<td>1698</td>
<td>1236</td>
<td>1526</td>
<td>502</td>
<td>3058</td>
<td>2292</td>
<td>1436</td>
</tr>
</tbody>
</table>

TABLE VIII: The length (in timestamp) of each event in Fig. 10a.

G. Test: Curated and Randomly Selected Detections

To further illustrate the performance of our CC-RCNN model, we present example detections of 5 long segments in Fig. 11 and Fig. 12. All detections are plotted on the top of the ground-truth. Fig. 11 shows the detection results of two curated time series segments, which are selected because of the impressive detection results. The detections of three randomly selected segments are presented in Fig. 12. Although they are not curated examples, we found the detection results still promising.

VII. Conclusion

Accurate event detection out of 1D times series seismic data is not only important, but also challenging. In this paper, we develop a novel event-wise detection method, DeepDetect, for 1D time series signals. Specifically, a cascaded architecture is developed to generate multi-scale proposals to detect events with various lengths. To take into account of the temporal correlation of time series data, we use atrous convolutions with different dilation rates to enrich features of individual proposals. To help with the optimization, we share parameters for branches built on top of multi-scale proposals. For event detection tasks in 1D time series signals, our model—DeepDetect—yield results significantly higher than that obtained using template matching, especially when the sizes of events are greatly different from one another. We also demonstrate the robustness of our new model to different sizes of the training dataset. To conclude, our new detection model, DeepDetect, yields high performance for the laboratory DeepDetect, yields high performance for a laboratory seismic dataset, therefore it has great potential for event detection in various seismic applications.

ACKNOWLEDGMENT

This work was co-funded by the Center for Space and Earth Science (CSES) at Los Alamos National Laboratory and the U.S. DOE Office of Fossil Energy through its Carbon Storage Program, and Institutional Support (LDRD) at Los Alamos.

REFERENCES


Fig. 10: Probability distributions from all detection branches. The input signals and the ground-truth are illustrated in (a). (b)-(h) exhibit probabilities of being an event for each location of the input signal, output by $D_3 - D_9$. Signals are denoted in blue light. Probabilities are denoted as solid lines of varying colors and the dotted red lines indicate the detection threshold of 0.5.
Fig. 11: Curated examples. The two segments are selected because we found the detection results are impressive.
Fig. 12: Randomly selected detection results. Our algorithm yields promising detections for most of the events, although there are some false-positive events obtained (e.g., one at Timestamp of 160,000 of Fig. 12(a)).
Responses to the Comments for Manuscript TGRS-2017-01371:
DeepDetect: A Cascaded Region-based Densely Connected Network for Seismic Event Detection

Yue Wu, Youzuo Lin, Zheng Zhou, David Chas Bolton, Ji Liu, and Paul Johnson

We thank Dr. Qingkai Kong, an anonymous reviewer, and the associate Editor for their valuable comments that help improve our manuscript. We have revised our manuscript to address their comments. The following is a detailed response to their specific comments.

The equation, table, and figure numbers in the format of “(X)” refer to those in the revised manuscript, while the ones in the format of “(0.X)” refer to those within this response.

Responses to the Reviewer 1’s comments

1. This is a very exciting and interesting paper that cast the earthquake detection problem to object recognition. The authors proposed the multi-scale structure to capture events with varying lengths. In order to reduce the degradation effect of a deep structure, they adapted the DenseNet as their network topology. The network is capable of generating different length of proposals and make a decision whether it is an earthquake or not as a classification problem. Once it is detected as an earthquake, it continues to assign a begin and end point to the proposals to localize the duration of the earthquake event by incorporate information from nearby proposals (taking into the consideration of temporal correlations). The proposed method was compared with the traditional method (template matching) on some experimental dataset, and the result looks really promising. This is a nice contribution to the seismology community, especially the detection community. As the seismic networks become dense, better detection algorithms are needed to pick out all the interesting signals. This paper takes the advantage of the deep neural network and object detection, and lays out a nice framework. Overall the paper is in good shape, and some places are not explained very well need clarification, but they are all very minor (see the minor comments below).

Answer:

We thank the reviewer’s questions and comments, and we have already addressed all those in numerical order below.

2. The dataset is still small 1000 events in total, even the test of different training percentage shows nice results (it is showing with more training data, the results are better), it is better to use some data augmentation method, or more experiment data (this one maybe hard, because it requires more man power to pick the positive cases) to increase the training dataset.

Answer:

We agree with the reviewer that using more data will be helpful to boost the performance. Currently, the labquake data we used in this paper is generated from the Rock & Sediment Mechanics Laboratory at Penn State University. Those events are manually picked. Having correct and high quality annotated events can be very expensive and labor intensive, since it requires experts with highly trained domain knowledge. As an example, it took a fully trained domain expert a whole week to obtain 500 events correctly labeled. In our current
manuscript, we have already deployed all the annotated datasets from Peen State University. With more annotated data sets available in the future, we are planning to report the performance of our detection algorithm in our future work.

On other hand, it is challenging to achieve promising results even when the dataset is not considerably large. We believe this is one of our major contributions in this work. We built our model with densely connected blocks as the backbone. We did extra experiments and found that the dense block based architecture indeed outperformed residual block based counterparts and those without skip connections.

3. Please show some of negative cases, and how you pick negative cases. This is usually the most difficult part in reality, there are so many different types of negative events, and how to find the most representative ones are important.

Answer:

Our domain expert only picked positive events (ground-truth). We determine positive/negative proposals by how much they overlap with the ground-truth, as described in Sec. IV-D. Those having Intersection over Union (IoU) above 0.5 with ground-truth are considered positive. Those having IoU below 0.3 are considered negative. Those with IoU \( \in [0.3, 0.5] \) do not contribute to the loss. We also notice that the expert may miss some positive events, so we apply label-dependent loss to compensate those mis-labeled positive events.

In reality, there can be many other types of negative cases caused by vehicles, human activities, etc. In our current scenario, we may have to hand-pick representative negative cases, or apply active learning techniques to automatically identify those cases. In labquake data, the negative cases are less misleading and they usually do not have huge amplitudes comparing to those picked positive events. So we simply treat proposals as negative ones if they do not overlap with the picked events above a certain threshold. We believe that our major contribution of this manuscript is the way we formulate the event detection problem where both detection and localization steps can be parameterized. Therefore, we use our labquake dataset, which is relatively clean, to demonstrate the feasibility of our methodology. We have great interests in dealing with real seismic data, and we will report the performance in future work, but this is out of the scope of the current manuscript.

4. This will continue from the above comment, the dataset used here is all from one experiment, it will be so different than the real cases. In the experiment data, the negative cases maybe more clean and consistent, but in reality, it is not. It is better to test the performance of the proposed model on some real datasets, and then compare with the template matching (there are many sort of dataset in the field these days published by other researchers maybe can be used as a benchmark dataset)

Answer:

We agree with the reviewer on the importance of testing our detection algorithms on the real cases. In fact, we are currently applying our methodology for some real seismic signals. We will report the resulting performance in the future.

5. In the paper, all the time series is in timestamp, please specify what is the sampling rate for the sensor, and what is the time span for the signal instead of just use how many timestamp,
since this will change when use different sampling rate. Please specify what are the sensors used in the tests, what is the sampling rate, and also all the figures with waveforms have no unit.

Answer:

Units of waveforms is in bits. The data acquisition system is 14-bit, and thus, the amplitude ranges from -16384-16384 bits. So the units do not corresponding to anything physical such as a displacement/velocity etc.

We have added the following discussion in our revised manuscript. Please see Page 7 for details.

The data used for our machine learning model comes from a frictional experiment in the Pennsylvania State University Rock and Sediment Mechanics Laboratory. The experiment was conducted using the double-direct shear configuration where two laboratory fault zones are sandwiched between two forcing blocks and sheared simultaneously.

Acoustical data is recorded from two broadband (.02-2MHz) piezoelectric transducers with a central frequency of 500kHz. The transducers are embedded within a 10×10 (cm$^2$) steel block and placed adjacent to the fault zone [Riviere et al., 2018]. Acoustical data is sampled continuously at 4MHz with a 14-bit Verasonics data acquisition system.

We hand-picked 1000 acoustic emissions (AEs) from the raw time series data between 2368.9-2369.7 (s). During this section of the experiment, the normal stress is at 3 MPa. Furthermore, this time corresponds to the first two seconds of the stick-slip cycle. For a single AE, we picked the beginning and end of the event based on the amplitude of the signal, duration of the signal and the characteristic shape of the signal. We use 800 events for training, 100 events for validating, and 100 events for testing.

6. Could you also show some false positive cases?

Answer:

In our results shown in Figs. 12, there are a few false-positive detections (e.g., one at Timestamp of 160,000 of Fig. 12(a)). We have added one sentence in our revised manuscript. To better illustrate the false-positive detection, we further provide two segments containing false-positive detections in Fig. 1 in this response letter. Detection #22, #23, #54, #55 are false detections and they represent what false detections look like with this dataset since there is no noise event with large amplitudes.

7. Page 1, line 18, 2nd column - typo ‘date sets’

Answer:

We have changed it to “datasets”.

8. Page 1, line 19, 2nd column and Page 2 line 52, 2nd column, I don’t think the reference 2 is in the unsupervised learning category, they use the earthquake signals and generating the fingerprints to detect more earthquakes, in this sense, they are not unsupervised.

Answer:
Figure 1: Two segments containing false positive detections. Predictions are denoted as green and red “●” and ground-truth are denoted by “×”. Detection #22, #23, #54, #55 are false detections.

We agree that reference [2] “Earthquake detection through computationally efficient similarity search” Yoon et. al should not be considered as an unsupervised method. We have added it to the supervised category in Sec. I and we have removed it from the unsupervised category mentioned in Sec. I & II.

9. Page 2, line 13, 2nd column, please explain (AP)@[0.50, 0.95], it is explained later, but it would be beneficial for readers to know

Answer:

We have added the following discussion in our revised manuscript. Please see Page 2 for details.

We use AP@[0.5, 0.95] as the evaluation metric. Average precision (AP) calculates the averaged maximum precision at each recall value. We calculate 10 APs with the intersection
over union (IoU) of [0.5:0.95:0.05] as the metric to identify true positive detections and then take the average.

10. In table I, please explain [conv3, 12]*6 means for the reader

   Answer:

   We have added the following description of our network in Sec. IV-A. Please see Page 4 for details.

   As illustrated in Table I, our DeepDetect is inspired by DenseNet. All convolution kernels in our network are 1D because of the input of 1D time series data. “Conv7, 64, /2” denotes using 64 $1 \times 7$ convolution kernels with stride 2. The same routine applies to max-pool and avg-pool. The brackets denote DenseNet blocks illustrated in Fig. 1. Specifically, we use “[conv3, 20] \times 6” to denote a DenseNet block where 6 convolution layers are applied, each followed by batch normalization and ReLU activation layers.

11. Page 4, line 53, 1st column, having strides 16-1024, is not clear; it is better to say explicitly, i.e. 16, 32, 64, 128, 256, 512, 1024. This is related with Page 4, line 40, 2nd column as well. ‘the shift between two adjacent anchors of D3 is 16 timestamps’, maybe more specific as, ‘the shift is 1/8 of the anchor size for each D3 - D9’

   Answer:

   We have changed the wording as suggested by the reviewer in our revised manuscript. Please see Page 4 for details.

12. Figure 2b, how do you know it is one event instead of two events? There is a chance that the 2nd signal maybe a smaller event. Please specify it why it is determined as one event.

   Answer:

   It could be two events in the time series. However, without locating the source on the fault, it is very difficult to know if what your are observing is indeed another event or simply a reflection from the previous event. This example in Figure 2b is probably one of the more complicated signals that we observe in our data. We are sure that there is at least one event in this segment. The criteria that we use for justifying an event is based on the amplitude of the event and the characteristics of the signal. The impulsive events typically reach a peak amplitude very quickly and decay exponentially back to some background level noise. However, there are some events in the signal that yield a broad rise time and broad decay (long-period events).

13. Page 5, line 26, 1st column, in the text it is saying the rates to be 4, 8, 12, but in the figure 3, it is 8, 16, 24? Same for line 38 as well

   Answer:

   We have corrected this inconsistency between our texts and Fig. 3. Please see Page 5 for details.

14. Page 5, line 27, 1st column, I didn’t see how the dilation rates are inspired by the amount of shifts of adjacent proposals. The shifts are from 16 to 1024 for different layers, but the rates here are 4,8,12.
**Answer:** The anchor sizes of D3-D9 are 128, 256, 512, 1024, 2048, 4096 and 8192; The shifts between adjacent proposals are 16, 32, 64, 128, 256, 512 and 1024 for D3-D9, respectively. With dilation rate 4, 8 and 12, the shifts between each atrous convolution node in D3 are 16*4=64, 16*8=128 and 16*12=192, which are 0.5, 1 and 1.5 of the anchor size of D3. In D4, those are 32*4=128, 32*8=256 and 32*16=512 which are 0.5, 1 and 1.5 of the anchor size of D4, so on and so forth.

As we state in the paper, we use these atrous convolution layers to capture contextual information. If the dilation rate is too small, features are similar. If the dilation rate is too large, features are irrelevant to the center proposal. Thus, we choose the dilation rate 4, 8 and 12.

15. **Page 5, line 36, 1st column, I didn’t understand ’0.5, 1, 1.5 of the anchor size’**

**Answer:**

Please see the answer to #14 above.

16. **Equation 9 and 10, the Gx, Gw should be the center and length of the ground truth?**

**Answer:**

To avoid the confusion, we have changed $G^*_x$ and $G^*_w$ in Eqs. (9) and (10) to $G_x$ and $G_w$ to denote predictions

\[
\begin{align*}
G_x &= P_w d_x + P_x, \\
G_w &= P_w \exp(d_w),
\end{align*}
\]

We also changed $G_x$ and $G_w$ in Eqs. (16) and (17) to $G^*_x$ and $G^*_w$ to denote ground-truth

\[
\begin{align*}
t_x &= (G^*_x - P_x) / P_w, \\
t_w &= \ln(G^*_w / P_w),
\end{align*}
\]

Please see Page 5&6 for details.

17. **Page 5, line 54, 2nd column, please explain more of the two conditional probability**

**Answer:**

We have added the following description of our network in Sec. IV-D. Please see Page 6 for details.

To obtain the optimal weight parameters $\alpha^*$ and $\beta^*$, Natarajan et al. [2015] gives

\[
\begin{align*}
\rho_{+1} &= P(\bar{Y} = -1 | Y = 1), \\
\rho_{-1} &= P(\bar{Y} = 1 | Y = -1), \\
\alpha^* &= \frac{1 - \rho_{+1} + \rho_{-1}}{2} , \\
\beta^* &= 1 - \alpha^*,
\end{align*}
\]

where $\bar{Y}$ denotes the observed label, $Y$ denotes the true label and the conditional probability $P$ denotes the probability of a sample being wrongly labeled.
18. *Page 6, line 18, 1st column, lambda is the regularization parameter? I didn’t understand, how the regression loss function have the effect of regularization?*

   **Answer:**

   We have corrected our texts to “λ controls the balance between \(L_{cls}\) and \(L_{regr}\).”

   Please see Page 6 for details.

19. *Page 6, line 47, 1st column, what’s P4-P8?*

   **Answer:**

   We have corrected it to \(D_3-D_9\). Please see Page 6 for details.

20. *Please explain Table II.*

   **Answer:**

   We have added the following discussion in our revised manuscript. Please see Page 6 for details.

   In our dataset, large events are rare comparing to small events. To account for this characteristic, we generate less proposals from large events than those from small events for training. In particular, Table II shows how we select the number of proposals from each detection branch.

21. *In figure 7, is the AP on test dataset?*

   **Answer:**

   Yes. When we use 500 training samples, 250 samples are used for testing. In particular, we split \(\{0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8\}\) of total samples as the training set, the validation and testing sets evenly split the rest of samples.

   Responses to the Reviewer 2’s comments

   1. *There’s a clerical error in section II: “Template matching is a detection method method that...“, delete the second word “method”.*

   **Answer:**

   We have deleted the second “method”
References


