DEBC Detection with Deep Learning

Ian E. Nordeng(✉), Ahmad Hasan, Doug Olsen, and Jeremiah Neubert

University of North Dakota, Grand Forks, ND, USA
{ian.nordeng,ahmadjarjis.hasan}@und.edu,
olsen@aero.und.edu, jeremiah.neubert@engr.und.edu

Abstract. This work presents a novel system utilizing state of the art deep convolutional neural networks to detect dead end body component's (DEBC's) to reduce costs for inspections and maintenance of high tension power lines. A series of data augmenting techniques were implemented to develop 2,437 training images which utilized 146 images from a sensor trade study, and a test flight using UAS for inspections. Training was completed using the Python implementation of Faster R-CNN's object detection network with the VGG16 model. After testing the network on 111 aerial inspection photos captured with an UAS, the resulting convolutional neural network (CNN) was capable of an accuracy of 83.7% and precision of 91.8%. The addition of 270 training images and inclusion of insulators increased detection accuracy and precision to 97.8% and 99.1% respectively.

Keywords: Convolutional neural networks · CNN · Inspections · Machine learning

1 Introduction

Infrastructure maintenance in the United States is a multi-trillion-dollar industry, of which the electrical grid makes up a significant portion [1]. Maintenance of the electrical grid poses substantial costs. To help reduce these costs, we developed a deep learning neural network capable of detecting the dead-end body component (DEBC) from high tension power lines to allow for further analysis of the part due to wear.

The DEBC is a full tension device that is used to attach the conductor to the power line structure while maintaining electrical current. This component is comprised of an outer aluminum sleeve with a four-bolt pad welded to one end, a steel forging with a steel eye, and an aluminum insert. The outer aluminum sleeve grips the aluminum strands of the power line, while the inner aluminum inserts grip the inner aluminum matrix core wires separately. The eye of the steel forging is connected to the insulator string on the dead-end tower or substation. Jumper connectors attach to the outer sleeve pad and are used to connect pairs of powerline conductors. To aid inspection and maintenance of the DEBC, the component was detected with a bounding box annotation allowing for segmentation and analysis of possible wear or failures of the DEBC.

Deep convolutional neural networks (CNN) were chosen for the task of detecting the DEBC as they have made a resurgence in visual recognition tasks in recent years,

P. Sharma and F.M. Bianchi (Eds.): SCIA 2017, Part I, LNCS 10269, pp. 248–259, 2017.
DOI: 10.1007/978-3-319-59126-1_21

overtaking other methods in image classification challenges [2]. This is further exemplified by the Pascal VOC challenge, a yearly challenge from 2005–2012, with the goal of recognizing objects from several visual object classes through a supervised learning process [4]. The challenge has commonly been used as a comparison between different object detection networks. Convolutional neural networks have consistently outperformed other methods and increased precision of detection in the Pascal Visual Object Classes (VOC) challenge [3].

Faster R-CNN was chosen for the purposes of aiding inspections as it was the state of the art in object detection available, as evidenced by the Pascal VOC 2007 challenge. The pretrained deep VGG16 model was implemented due to its high precision and public availability, as shown in [5]. All training in this work was completed using a Titan X GPU.

1.1 Background

To determine the most effective object detection CNN available, four different object detection CNN's were considered including R-CNN, SPPnet, YOLO, and Faster R-CNN. Due to the post processing focus of the proposed algorithm, accuracy and precision were the main considerations for each network. Table 1 lists a direct comparison of each network on the Pascal VOC 2007 dataset as represented by the mean average precision (mAP) attained on the challenge by each network.

Table 1. mAP of each detection network considered.

CNN considered	mAP in Pascal VOC 2007
R-CNN	58.5%
SPPnet	60.9%
YOLO	63.4%
Faster R-CNN	73.2%

R-CNN was the first successful object detection algorithm utilizing a CNN, and increased mAP of the previous state of the art method by over 30% [6]. This was done by utilizing region proposals. Region proposals are specific regions in the image determined as an object, provided by a separate algorithm, and performing a convolutional network forward pass for each proposed region. Although originally successful, there are many drawbacks to the region-based convolutional neural network. First, training is a multi-staged pipeline requiring finetuning the CNN on object proposals generated separately, then fitting a support vector machine (SVM) to the CNN features, and performing bounding box regression. Second, training is expensive in both hard drive space and time spent during training. The SVM and bounding box regression training requires storing features extracted from each object proposal to disc which may require storing hundreds of gigabytes of data. Third, detection is slow as features are extracted from each object proposal in each test image. Methods considered below improve both speed and precision over this implementation.

Spatial Pyramid Pooling (SPP-net) was proposed to speed up R-CNN by sharing computation [7]. SPP-net first computes a convolutional feature map for the entire input image, then classifies each object proposal using a feature vector from the shared feature map. Features are extracted through maxpooling the portion of the feature map inside the proposal into a fixed output size. Training is still multi staged, as it must extract feature vectors, fine tune the network with log loss, train a SVM, and finally fit bounding box regressors. SPPnet cannot update the convolutional layers preceding the spatial pyramid pooling limiting accuracy of deep networks.

You only look once (YOLO), is a real-time object detection CNN [8]. This method applies a single neural network to the full image at test time to provide global context, divides the image into equally spaced regions, and predicts bounding boxes and probabilities for each region. This method provides a fast object detection that runs in real time, up to 45 FPS, for the more computationally expensive and accurate model. As stated in [8], this network provides a mAP of approximately 10% less than Faster R-CNN.

Faster R-CNN is the latest improvement over R-CNN and introduced Region Proposal Networks (RPN) that share convolutional layers with the object detection network [5]. The region proposals are created by adding two additional convolutional layers. The first layer encodes each convolutional feature map position into a feature vector. The second layer outputs an objectness score and regressed bounds for k region proposals, relative to various scales and aspect ratios, at each convolutional map position. These added layers create a fully-convolutional network that can be trained end-to-end for the task of generating detection proposals. Faster R-CNN also developed a training scheme that alternates between fine-tuning for the region proposal task, and fine-tuning for object detection with the proposals fixed. Faster R-CNN was chosen for our purposes as it achieved the highest mAP of all methods considered at 73.2%.

2 Procedure

To properly train Faster R-CNN for detection of DEBC's, several tasks were completed with the following processes. The Python version of Faster R-CNN was obtained via [5] utilizing the VGG16 model as explained in Sect. 2.1. Section 2.2 expands on data collection of DEBC's. Section 2.3 provides methods for data augmentation to enhance the training dataset. Section 2.4 details the creation of the ground truth annotations for the training data. Section 2.5 provides the methods for collection of test data. Section 2.6 explains the process for training the different networks considered.

2.1 Faster R-CNN

The Python reimplementation of Faster R-CNN was obtained from [5]. As previously mentioned, Faster R-CNN incorporates a small region proposal network that shares a common set of convolutional layers with a standard detection network and was built using the Caffe framework [9]. This region proposal network takes an image of any size as input, and outputs rectangular object proposals with objectness scores, or in other

words, a measurement of belonging to a set of object classes *vs*. the background. Training the region proposal network layers is accomplished through assigning a binary object or not object to anchors that are predicted from k region proposals. These anchors are assigned as positive while training when they either have the highest Intersection-over-Union (IoU) overlap with a ground truth box or IoU overlap greater than 0.7 with any ground truth box. Non-positive anchors did not contribute to training. The loss function for an image,

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}\left(t_i, t_i^*\right), \tag{1}$$

is minimized to generate a trained network. In the loss function i is the anchor index in a mini batch, p_i the predicted probability of the i'th anchor being an object. p_i^* represents the ground-truth label and is set to 1 for a positive anchor and 0 if negative. t_i provides the vector representation of the four parameterized coordinates of the predicted bounding box with t_i^* the ground-truth box associated with the positive anchor. The classification loss L_{cls} represents the log loss over the two object *vs*. non object classes. The regression loss $L_{reg}\left(t_i, t_i^*\right) = R\left(t_i, t_i^*\right)$ where R represents the robust loss function. The L_{cls} and L_{reg} functions are normalized with N_{cls}, N_{reg}, and the balancing weight λ. The classification of the region can then be performed once the regions for the object detected are determined. In this work, the VGG16 model was used to perform this classification.

The VGG16 model utilizes a total of 16 convolutional and max pooling layers with max pooling occurring after every three convolutional layers [10]. After each convolutional layer a ReLU layer was applied to increase nonlinearity. The response-normalized activity $b_{x,y}^i$ is given by

$$b_{x,y}^i = \frac{a_{x,y}^i}{\left(k + \alpha\sum_{j=\max\left(0,i-\frac{n}{2}\right)}^{\min\left(N-1,i+\frac{n}{2}\right)}\left(a_{x,y}^j\right)^2\right.} \tag{2}$$

in which $(a_{x,y}^j)$ denotes the activity of a neuron computed by applying kernel I at (x, y), the sum runs over n adjacent kernel maps, and N represents the total number of kernels in the layer. The last three layers of the VGG16 CNN are the fully connected layers to perform the classification.

A Titan X GPU was used to train Faster R-CNN with the complex VGG16 network, which required ∼11 Gb of GPU memory. Training was performed using the alternating optimization method per [5]. This method performed a 4-step training to learn shared features between the region proposal network with a separate detection network. First, the RPN was trained as above. Second, the detection network was trained separately without sharing layers. Third, the detection network was used to initialize the region proposal network training but fixed the shared layers, fine-tuning only the region proposal network layers. The final stage shared the convolutional layers, keeping them fixed, and fine-tuned the fully connected layers, which formed a unified network.

2.2 DEBC Data Collection

The original data was provided from two local businesses. The first provided data from a sensor trade study [11], and the second a UAS test flight. The trade study collected data to determine the optimal camera sensor, viewing angle, and distance for a human to be able to identify potential maintenance concerns.

In the trade study, the cameras used to collect the data include the Sony NEX 7, and a Sony α6000 sensor converted to perform as a multispectral camera. The converted Sony α6000 sensor provided 700–800 nm wavelength light along with the standard visible light. Each image was of size 6000 × 4000 pixels. Two different DEBC's were attached to a forklift and lifted to the test height ranging from 4 m to 12 m from the camera height, and tested under various weather conditions. A total of 111 images from the sensor trade study were provided.

From the UAS test flight, images were collected on live high voltage power lines. The UAS flew approximately 15–20 m from the DEBC's imaged. A sensor comparable to the Sony NEX 7 from Field of View was used for data collection. For our purposes, only images with the DEBC within view were considered. From the UAS test flight, 30 training images were collected.

With a total of 146 images (Fig. 1), the training data set required a large amount of data augmentation to be performed to allow for a sufficiently large training set. As a comparison, the Pascal VOC training dataset provides close to 5,000 images per class to be considered. An additional 270 images from another inspection flight were later included and unaltered to improve the network accuracy. In addition to the 270 images, another class of images considering insulators was added to help the network differentiate between insulators and DEBC's.

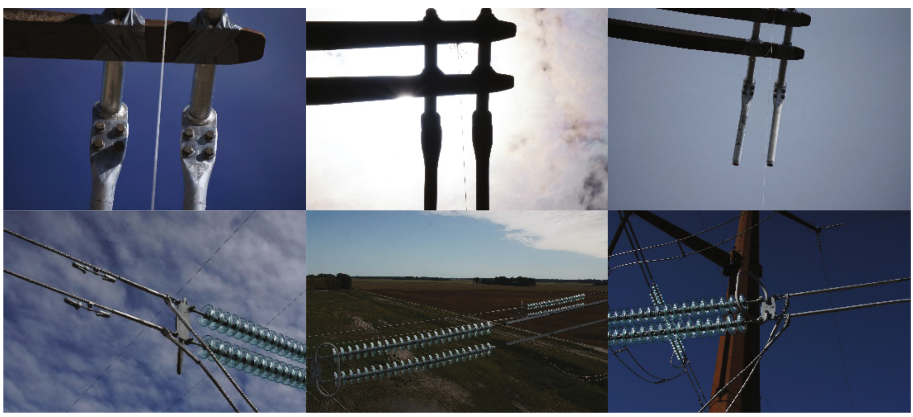
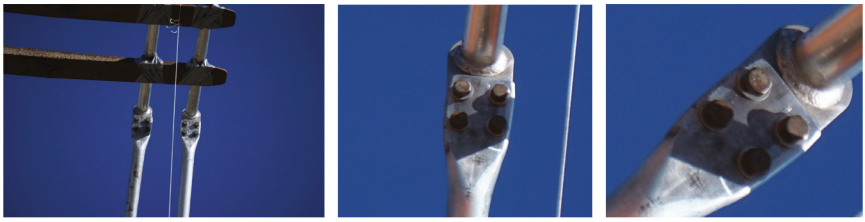


Fig. 1. Example images from training data. Top images from sensor trade study, bottom images from test flight.

2.3 DEBC Data Augmentation

To generate enough data to properly train the network to accurately detect the DEBC, a series of simple image processing techniques were performed (Fig. 2). All image processing methods were completed using OpenCV functions [12].



(a) (b) (c)

Fig. 2. Sample augmented images. (a) Original image. (b) Cropped image of left DEBC. (c) Image rotated 45º.

The first method involved manually cropping the DEBC from each original image. Due to the Faster R-CNN algorithm automatically resizing all input images to 1000 pixels on the larger side, and 600 pixels on the smaller side, the cropped images allowed for increasing the robustness of the network to differences of scale. The cropped images were also used for multiple data augmentations below.

To account for various viewing angles the DEBC may be oriented, each cropped image from before was manipulated to create a series of rotations. Rotations were performed using an SO(3) rotation matrix in degrees. The rotation matrix as

$$Rotation\,Matrix = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

was applied to the image. Rotations were applied using inverse warping where each intended pixel location in the rotated image is computed, then the corresponding location in the original image is sampled. The cropped images were rotated from their original position in 15º intervals to a total of 60º. The images were then cropped to remove the resulting black corners of the image from the rotations.

To account for possible out of focus or grainy images, two different morphological operations were performed on the images. These morphological operations include a slight dilation and erosion. This process is done by convolving a kernel (β) over the image I. β has a defined anchor point at the center of the kernel. For dilation, kernel β is convolved over the image and the maximal pixel value overlapped by β is computed and replaced by the image pixel in the anchor points position with that maximal value. This causes bright regions within an image to expand. Erosion is done similarly but instead uses the minimal pixel value for the anchor point causing bright regions. For our purposes, β was chosen to be of size [3 × 3] with only a single pass for slight

erosion and dilation operations. Each of the above images were then flipped horizontally.

The last method cropped 1000 × 1000 pixel sized patches in a raster scan pattern with 50% overlap from the original 6000 × 4000 images. This technique was performed to create translations of the DEBC, as well as allow for edge cases where the DEBC would be only partially visible due to being truncated at the edge of the cropped image. The 50% overlap was used to ensure parts of the image would always be visible. Lastly, the total images had to be manually sorted to remove images without the DEBC visible.

With all image processing techniques completed, a total of 2,437 images were developed. With many images to train the network now available, the data needed to be annotated for the CNN to train on the object in each image to be detected.

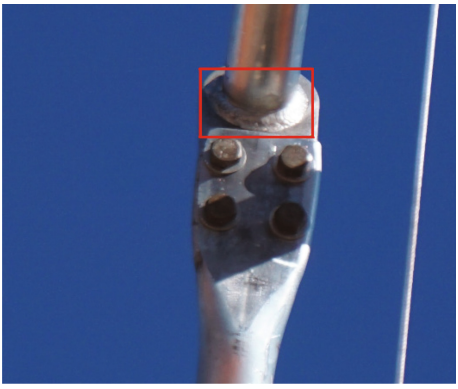


Fig. 3. Example image with ground truth bounding box annotation.

Fig. 4. Example Image from validation set.

2.4 Annotations

Training a Faster R-CNN model required all cases of the intended object categories to be annotated. For our purposes, one class was considered outside the background class, a catch all for all non-defined objects, as the DEBC. Each annotation was created to encompass a portion of the DEBC while limiting all other features. A second class which considered the insulators was annotated and added to the training set later. With Faster R-CNN developed to train on the Pascal VOC dataset, the annotations performed matched the format using the LabelImg software [13]. The Pascal VOC format stores each bounding box annotations location, class, and image location on file in xml format. All annotations were manually selected and a sample of an annotation is provided in Fig. 3.

2.5 Test Data

Images to test the network performance were needed to evaluate the trained networks. Figure 4 provides sample images that were utilized and provided by the company which collected the test flight training data, and provided a newer set of true inspection images for the DEBC's. This test data included 111 images which include 115 total DEBCs taken from a much closer range and viewing angle in which the DEBC was easier to view. These test images were kept separate from the training data.

2.6 Training

The networks evaluated were fine-tuned from an ImageNet pre-trained VGG16 model over differing numbers of iterations and numbers of images following the methods in [14]. The learning rate was set as 0.0001 for 60k mini-batches and 0.00001 for the next 20k mini-batches, momentum as 0.9, and weight decay as 0.0005 as provided by the VGG16 model. The numbers of iterations varied from 40,000 to 150,000, often alternating the higher number of iterations in the first and third stage, and the lower number of iterations in the second and fourth. These alternating numbers of iterations were done as the default iterations were set to alternate from 80,000 to 40,000. The most visually accurate networks based on the number of iterations run were chosen for a full evaluation of the network. Time training the network was heavily dependent on the number of iterations, but took approximately three to seven days of training.

3 Results

The 111 test images were used to evaluate several different trained networks while varying both the number of iterations the network was trained with, and the number of training images. ROC and Precision-Recall curves were developed for each trained network to determine network accuracy (Fig. 5). Data for the curves was generated by setting the threshold for detection to the value of 0.1, and storing all detections and confidence intervals.

With all data available, true positives, true negatives, false positives, and false negatives were recorded while varying the threshold from 0.1 to 1 in 0.05 intervals. Figure 6 provides examples of how the network could accurately locate the DEBC in a variety of positions and poses, including one image where most the DEBC was outside the image. True positives were classified as matches if the detection appeared visually correct allowing for the user to easily view and evaluate the DEBC condition within the bounding box. True negatives were only considered from the list of false positives. As the threshold increased and the false positives were no longer detected, they became true negatives. False positives were defined as any detections that were not of a DEBC. False negatives were tallied for any DEBC not detected in the dataset. Both ROC and Precision Recall curves were developed by calculating the recall/true positive rate (TPR), the false positive rate (FPR), and precision as $TPR = \frac{TP}{TP+FN}$, $FPR = \frac{FP}{TN+FP}$, and $Precision = \frac{TP}{TP+FP}$ respectively. The ROC curve for all three networks considered

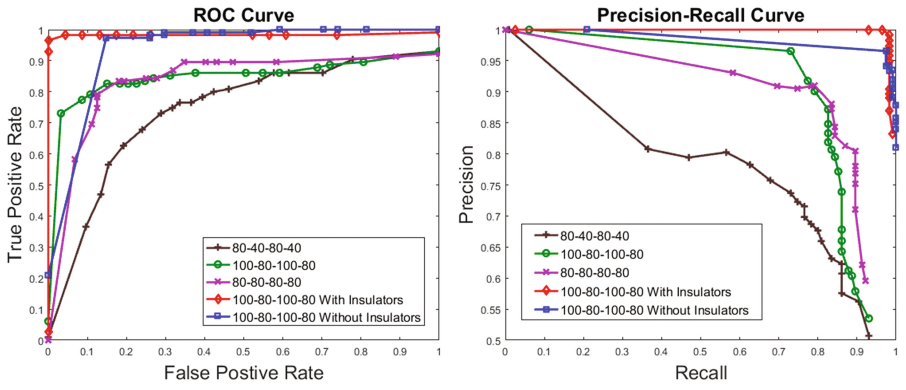


Fig. 5. (a) ROC curve of five different networks, numbers list the iterations of each stage of training for that network (in thousands). The curve with insulators includes additional images and a separate class considering insulators, the curve without insulators includes the additional images, but not the separate insulator class. (b) Precision Recall curve for the same five networks in (a)

was plotted using the TPR, and FPR, whereas the Precision Recall curve was plotted using the precision and recall at each threshold value. After finding a high rate of false positives due to detecting insulators as DEBC's, an additional 270 images were included in the network with insulators as a separate class. The network was retrained with the more successful number of iterations, 100,000, 80,000, 100,000 and 80,000.

As per [15], the network trained with 100,000, 80,000, 100,000, and 80,000 iterations dominates the ROC space of the first three trained networks at higher confidence interval thresholds as evidenced in Fig. 5. Taking the closest point to a TPR of 1 and FPR of 0 in the ROC curve while closer to a precision of 1 and recall of 1 in the precision recall curve, a threshold of 0.9 was found using the network trained with 100,000, 80,000, 100,000 and 80,000. From this data, the accuracy, $acc. = \frac{TP+TN}{Total}$, was also found. Accuracy was determined as 83.7% while maintaining a precision of 91.8%. A second point was also considered with a threshold of 0.85, but resulted in a slightly lower precision of 90.01% and the same accuracy. The confusion matrix at the 0.9 threshold is listed in Table 2.

Inclusion of the additional 270 images, along with the new insulator class, substantially improved the detections of the DEBC. Adding the 270 images increased the TPR, while the addition of the insulator class decreased the FPR on the ROC curve. The Precision Recall curve demonstrates inclusion of the additional images and including the additional insulator class dominates in the ROC space. Due to a preference for higher recall, the F_2 measure was calculated. The highest score was found as $F_2 = 0.6168$ at a threshold of 0.85. The accuracy and precision of this network at a threshold of 0.85 is 97.8% and 99.1% respectively with the confusion matrix for this threshold listed in Table 3.

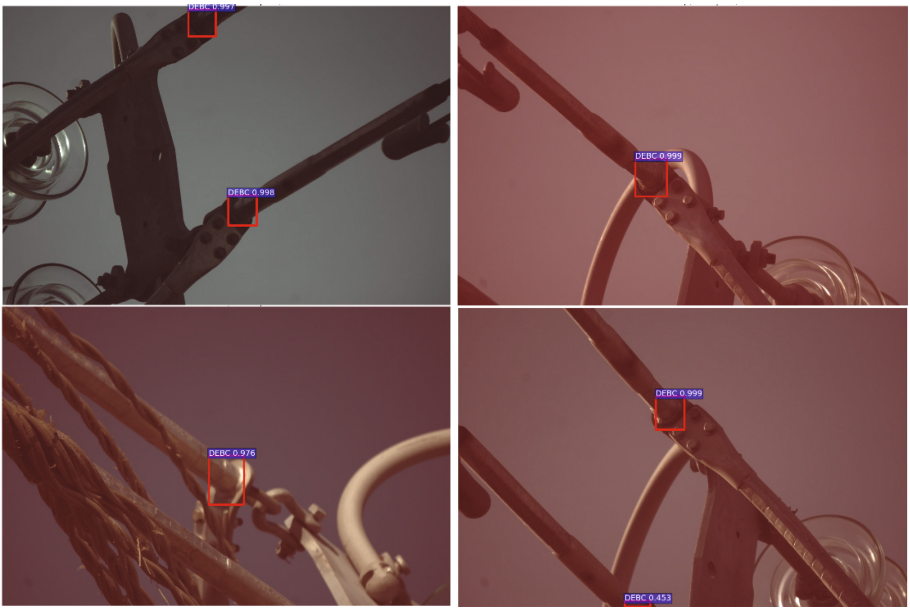


Fig. 6. Example results of successful DEBC detections.

Table 2. Confusion matrix for network 100, 80, 100, 80 (in thousands) with threshold 0.9.

	Predicted negative	Predicted positive	Total
Actual negative	TN: 85	FP: 8	93
Actual positive	FN: 26	TP: 89	115
Total	111	97	208

Table 3. Confusion matrix for network 100, 80, 100, 80 (in thousands) with additional images and insulator class, threshold 0.85.

	Predicted negative	Predicted positive	Total
Actual negative	TN: 22	FP: 1	23
Actual positive	FN: 2	TP: 113	115
Total	24	114	138

Failures fell mostly into two categories, false positives with high confidence, and false negatives (Fig. 7). False positives occurred primarily due to insulators in the background, and are believed to occur due to a lack of insulators in the training set. Inclusion of the additional images and the insulator class diminished these false positives. False negatives were attributed to both variation in lighting conditions and viewing angle, likely due to a lack of variability in the original training data set. Inclusion of additional training images helped in these detections.

(a) (b)

Fig. 7. Failures in detection. (a) Occluded DEBC not detected. (b) False positive DEBC detection

4 Conclusion

In this work, deep convolutional neural networks were applied and evaluated based on detection of DEBC's from dead end high tension power lines using Faster R-CNN. It was demonstrated that increasing the number of iterations for each of the four stages of training to 100,000, 80,000, 100,000 and 80,000 increased detection accuracy. Including 270 additional images, subsequently increasing the variability of the training set, increased the TPR. After finding a common object that caused a high number of false positives, inclusion of that object as a separate class decreased the false positive rate.

Future improvements to the network are suggested. Per [16], removing difficult training images can increase the precision. The training data from the sensor trade study provided several images where the DEBC was difficult to see due to situations such as the sun in the direct background. Most false positives were found to be other round shaped objects in the image. Additional classes and training images may help to reduce these false positives further. Increasing the annotation size for the DEBC to include more features, for example the four bolts on the DEBC, may also help reduce or eliminate this problem. Additionally, adding more classes can increase the utility of the network.

References

1. McNichol, E.: It's time for states to invest in infrastructure (2016). http://www.cbpp.org/research/state-budget-and-tax/its-time-for-states-to-invest-in-infrastructure
2. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. In: NIPS 2012
3. Girshick, R.: Fast R-CNN object detection with Caffe. From Caffe CVPR presentation (2015). http://tutorial.caffe.berkeleyvision.org/caffe-cvpr15-detection.pdf
4. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes (VOC) challenge. IJCV **88**(2), 303–338 (2010)

5. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS 2015. https://github.com/rbgirshick/py-faster-rcnn
6. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR 2014
7. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 346–361. Springer, Cham (2014). doi:10.1007/978-3-319-10578-9_23
8. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. arXiv 2016
9. Jia, Y., et al.: Caffe: Convolutional Architecture for Fast Feature Embedding. In: ACM 2014. http://caffe.berkeleyvision.org/installation.html
10. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recongition. In: ICLR 2015
11. Lemler, K., Heichel, J., Dvorak, D.: Powerline Sensor Trade Study Sensor Test Plan (2016)
12. OpenCV Dev Team: OpenCV Documentation (2016). Accessed http://opencv.org/documentation.html
13. Tzutalin. LabelImg. Git code (2013). https://github.com/tzutalin/labelImg
14. Beaucorps, P.: Train Py-Faster-RCNN on another dataset (2015). https://github.com/deboc/py-faster-rcnn/tree/master/help
15. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: ICML 2006
16. Girshick, R.: Fast R-CNN. In: ICCV 2015