Received 25 June 2018; revised 6 July 2018, 17 August 2018 and 9 October 2018; accepted 8 November 2018. Date of publication 11 January 2019; date of current version 25 March 2019.

Digital Object Identifier 10.1109/JPETS.2018.2881429

Intelligent Monitoring and Inspection of Power Line Components Powered by UAVs and Deep Learning

VAN NHAN NGUYEN[1,2], ROBERT JENSSEN[1], (Member, IEEE), AND DAVIDE ROVERSO[2]

[1]The UiT Machine Learning Group, UiT/The Arctic University of Norway, 9019 Tromsø, Norway
[2]Analytics Department, eSmart Systems, 1783 Halden, Norway
CORRESPONDING AUTHOR: V. N. NGUYEN (nhan.v.nguyen@uit.no)

This work was supported in part by the Research Council of Norway [RCN NÆRINGSPHD Grant 263894 (2016–2018) on Power Grid Image Analysis] and eSmart Systems.

ABSTRACT In this paper, we present a novel automatic autonomous vision-based power line inspection system that uses unmanned aerial vehicle inspection as the main inspection method, optical images as the primary data source, and deep learning as the backbone of the data analysis. To facilitate the implementation of the system, we address three main challenges of deep learning in vision-based power line inspection: (i) the lack of training data; (ii) class imbalance; and (iii) the detection of small components and faults. First, we create four medium-sized datasets for training component detection and classification models. Furthermore, we apply a series of effective data augmentation techniques to balance out the imbalanced classes. Finally, we propose the multi-stage component detection and classification based on the Single Shot Multibox detector and deep Residual Networks to detect small components and faults. The results show that the proposed system is fast and accurate in detecting common faults on power line components, including missing top caps, cracks in poles and cross arms, woodpecker damage on poles, and rot damage on cross arms. The field tests suggest that our system has a promising role in the intelligent monitoring and inspection of power line components and as a valuable addition to smart grids.

INDEX TERMS Intelligent monitoring, power line inspection, vision-based power line inspection, deep learning, UAVs, smart grids.

I. INTRODUCTION

TO PREVENT power outages, electric utilities regularly perform visual inspections on their power grids to plan for necessary repair or replacement work. These inspections have typically been carried out using traditional methods such as foot patrol and helicopter-assisted surveys, which are typically slow, expensive, and potentially dangerous. In recent years, many researchers have been seeking to develop fast and accurate methods for automatic autonomous power line inspection. Unfortunately, to the best of our knowledge, no fully automatic autonomous power line inspection systems have been developed.

The work presented in this paper aims to realize fast, accurate, and safe automatic autonomous power line inspection, and is part of an ongoing effort to exploit recent advances in Deep Learning (DL) and Unmanned Aerial Vehicle (UAV) technologies for this purpose. In our previous work [1], we conducted a review of the main power line inspection tasks, the existing power line inspection methods, and the potential data sources for power line inspection. Based on that, we proposed a novel automatic autonomous vision-based power line inspection concept that uses UAV inspection as the main inspection method, optical images as the primary data source, and deep learning as the backbone of the data analysis. To move forward, we identified six main challenges of DL vision-based UAV inspection: the lack of training data; class imbalance; the detection of small components and faults; the detection of previously unseen components and faults; the detection of power lines in cluttered backgrounds; and the lack of metrics for evaluating inspection performance.

In this paper, we take this concept further and address the first three challenges by creating four medium-sized datasets for training component detection and classification models, by applying a series of effective data augmentation techniques to balance out the imbalanced classes, and by utilizing multi-stage component detection and classification based on Single Shot multibox Detector (SSD) [2] and deep Residual Networks (ResNets) [3] to detect small components and faults.

Having addressed the first three challenges, we build a basic automatic vision-based power line inspection system with two custom-built UAVs and five DL-based models for data analysis and inspection. The remaining three challenges are left for future work which involves facilitating self-driving UAVs with power line detection as well as advancing automatic inspection at large-scale with a wide range of faults to realize fully automatic autonomous power line inspection.

The work presented in this paper does, in our opinion, demonstrate the potential role and the importance of automatic autonomous power line inspection in the intelligent monitoring of power grids. High-speed UAVs equipped with sensors, cameras, and DL vision-based models for navigation and data analysis can automatically navigate along power lines to collect data for offline inspections and perform online inspections to quickly identify faults on both power line components and the power lines themselves.

The remainder of the paper is structured as follows: Section II presents background knowledge and relevant related work on UAV inspection, vision-based inspection, and DL-based classification and detection models, before we describe our proposed automatic autonomous vision-based power line inspection concept in Section III. Next, in section IV, we present in detail our proposed approaches. Then, in Section VI, we present experimental results and discuss the potential of our proposed system in the intelligent monitoring of power grids. Finally, in Section VII, we conclude the paper with a summary and an outlook for the future of the field.

II. BACKGROUND AND RELATED WORK

Power lines are traditionally inspected at regular intervals by foot patrol or by helicopter-assisted surveys. In these inspection methods, a team of inspectors is sent out traveling either on foot or by helicopter to collect data for offline inspections and for visual inspection of the power lines. To improve inspection speed, accuracy, and to reduce inspection costs, a considerable amount of research has been conducted in order to automate vision-based power line inspection.

A. UAV INSPECTION

In recent years, advances in battery and fuel cell technologies [4], sensors, and UAV components [5] have significantly improved the feasibility of UAV-based power line inspection. However, the current applications of UAVs in power line inspection are still facing many unsolved challenges.

Deng *et al.* [6] identified three main challenges that prevent UAV inspection from daily services. The first challenge is the application modality. A single UAV typically can only cope with a specific power line inspection task; thus, multi-UAVs are usually required to perform full inspections. The second challenge is the lack of communication and control systems, and the third challenge is the gap between data collection and data analysis. To address these challenges, the authors proposed a cooperative power line inspection paradigm using multi-platform UAVs: a fixed wing UAV for long-distance brief inspection, a multi-rotor UAV for short-distance thorough inspection, and a tethered muti-rotor UAV for communication relay. Field tests suggested that the proposed approach outperforms traditional inspection methods (e.g., foot patrol).

B. VISION-BASED INSPECTION

With recent advances in deep learning for computer vision, cameras, and sensors, vision based power line inspection is drawing increasing attention from the power industry. The main reason why vision-based inspection is popular is that it can cover a wide range of faults on a single inspection [1].

Reviews of different data sources for vision-based inspection and existing vision-based inspection systems can be found in [1] and [7]. Based on the reviews, the current authors proposed optical images collected by UAVs as a potential data source for vision-based inspection because (i) they are easy to collect; (ii) relatively easier to analyze than the other reviewed data sources, while; (iii) providing enough information for detecting a wide range of common faults on both power line components and the power lines themselves.

C. DL-BASED CLASSIFICATION AND DETECTION MODELS

In the past few years, Convolutional Neural Networks (CNNs) [8], which are a special kind of neural networks designed to take advantage of the 2D structure of image data, have been advancing the state of the art of many computer vision applications, such as image recognition and object detection. In this section, we briefly describe the underlying concept of CNNs and summarize some of the most well-known CNN architectures for image classification as well as CNN-based frameworks for object detection.

The four key ideas behind the successes of CNNs in processing image data are local connections, shared weights, pooling, and the use of many layers [9]. A CNN for image classification is typically composed of three types of layers: convolutional layers, pooling layers, and fully-connected layers.

Convolutional layers are the fundamental component of CNNs which leverages the three main ideas that make CNNs powerful: local connectivity, parameter sharing and equivariant representations [10]. A convolutional layer accepts a volume I of size $[W_I, H_I, D_I]$ as input and outputs a volume O of size $[W_O, H_O, D_O]$. The convolutional layer is composed of several convolution *kernels* K (often called *filters*). Each neuron in the output volume looks at a rectangular region in the input volume. The rectangular region is referred to as the neuron's *receptive field* in the previous layer, and the size of the region is often called the *filter size* [11]. The filters are slided across the input volume I with *stride* S to compute dot products to produce *activation maps*:

$$O_K(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n). \tag{1}$$

In practice, many deep learning libraries implement an alternative function called the cross-correlation:

$$O_K(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n). \tag{2}$$

According to [10], pooling layers ‘‘replace the output of the net at certain locations with summaries statistic of the nearby outputs’’. There are many pooling functions that can be used in pooling layers, such as max pooling, average pooling, and $L2$-norm pooling. However, in practice, it is recommended to use the max pooling function, which takes a rectangular region P as input and outputs the maximum value of the elements in the region. The pooling function is slided across the input volume I with stride S to compute activation maps:

$$O_P(i,j) = \max_{m,n \in P(i,j)} I(m,n). \tag{3}$$

Pooling layers in CNNs serve two main purposes. First, pooling introduces invariance to small translations in the input. The second is that pooling reduces the amount of parameters and computation in the network by progressively reducing the spatial dimension of the input volume. However, it has been shown that max pooling can simply be replaced by a convolutional layer with increased stride [12].

Fully-connected layers, which are typically responsible for high-level reasoning in CNNs, are composed of neurons that are connected to all activations in the previous layer, as seen in regular neural networks.

Many well-known deep CNNs, such as AlexNet [8] and VGGNet [13], are formed by simply stacking up many convolutional layers, pooling layers, and fully-connected layers. In those deep CNNs, the information flowing through the network passes through many stages of multiplication; therefore, the gradients are needed to be back-propagated though many stages during training. This causes the gradients to either vanish or explode. The exploding gradients problem can be addressed easily by, for example, applying gradient clipping. The vanishing gradients, on the other hand, are quite hard to overcome. When the gradients vanish, the learning either becomes very slow or stops working. This issue is historically known as one of the main challenges of training very deep CNNs. An example of the vanishing gradient problem's cause is the use of saturated activation functions such as the hyperbolic tangent (*tanh*) or the logistic sigmoid [14]. In modern CNNs, it is recommended to use non-saturated activation functions, which typically suffer less from the vanishing gradient problem, such as the Rectified Linear Units (ReLU), as alternatives to the hyperbolic tangent or logistic sigmoid [15]. The ReLU is defined as

$$ReLU(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0. \end{cases} \tag{4}$$

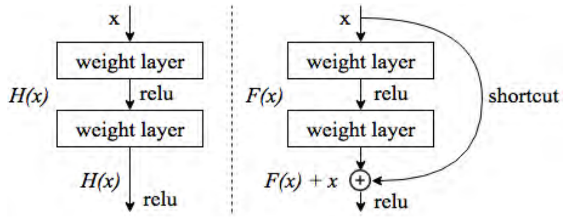

FIGURE 1. Standard CNNs (left) vs ResNets with shortcut connections (right). $H(x)$ is the underlying mapping. $F(x) = H(x) - x$ is the residual mapping adopted by ResNets.

For more details on the underlying concept of CNNs and their existing challenges, we refer the interested reader to [9], [10], and [11].

1) RESNET

With the increasing complexity of image classification problems, deeper CNNs are typically required. However, as mentioned above, deep CNNs constructed simply by stacking up many layers are very difficult to train due to the notorious problem of vanishing/exploding gradients. To ease the training of deep CNNs, Residual Networks (ResNets) were proposed [3]. ResNets add ‘‘shortcut’’ connections to the standard CNNs layers to allow the gradient signal to travel back directly from later layers to early layers (See Fig. 1). The ‘‘shortcut’’ connections allowed the authors of the ResNets to successfully train very deep CNNs with 50, 101, and even 152 layers.

2) FASTER R-CNN

Inspired by the successes of CNNs in image classification, Faster R-CNN (Region-based Convolutional Neural Network) was proposed to solve a more challenging task of object detection. Faster R-CNN is a single, unified network which performs object detection via two main steps: region proposal and region classification. First, a base network (e.g., ResNet [3]) is utilized to extract features from images. Next, the extracted features are fed into a Region Proposal Network (RPN) to find proposals. Then, a CNN-based classifier is applied on top of the extracted feature maps to classify the proposals and refine their bounding boxes, which are rectangles that enclose a single detected object. Finally, post-processing is used to refine the bounding boxes and eliminate duplicate detections. Faster R-CNN is very accurate; however, it is quite slow.

3) R-FCN

R-FCN (Region-based Fully Convolutional Network) is an accurate and efficient object detection framework proposed to address existing issues of region-based detectors such as Fast/Faster R-CNN [16]. Instead of applying costly per-region sub-network hundreds of times, R-FCN adopts a fully convolutional architecture with almost all computations shared across the entire image. To address the dilemma between translation-invariance in image classification and translation-variance in object detection, R-FCN proposes novel position-sensitive score maps which allow fully convolutional networks to effectively and efficiently perform both classification and detection in a single evaluation. With those novel improvements, R-FCN can run at 2.5-20 times faster and achieve higher accuracy than the Faster R-CNN counterpart.

4) YOLO

YOLO (You Only Look Once) is a real-time object detection framework that directly predicts bounding boxes and class probabilities with a single network in a single evaluation [17]. To achieve this, YOLO unifies region proposal and region classification into a single neural network and, according to the authors, ‘‘frames object detection as a regression problem to spatially separated bounding boxes and associated class probabilities’’. YOLO divides the input image into an $S \times S$ grid. Each grid cell predicts B bounding boxes, confidence score for those boxes, and C conditional class probabilities. With a unified architecture, YOLO is extremely fast; it processes images in real-time. However, YOLO is not state-of-the-art in terms of accuracy.

5) SSD

SSD (Single Shot MultiBox Detector) improves YOLO by adding a series of modifications: (i) a small convolutional filter is utilized to predict object classes and offsets in bounding box locations; separate predictors (filters) are employed for predicting object at different aspect ratios; predictions are performed at multiple feature maps from the later stages of a network to enable detection at multiple scales [2]. These modifications make SSD both faster and more accurate than the YOLO counterpart.

III. THE PROPOSED AUTOMATIC AUTONOMOUS VISION-BASED POWER LINE INSPECTION CONCEPT

With the aim of utilizing recent advances in deep learning and UAV technologies to realize fast, accurate, and safe power line inspection, we propose a novel automatic autonomous vision-based power line inspection concept that uses UAV inspection as the main inspection method, optical images as the primary data source, and deep learning as the backbone of the data analysis.

A. CUSTOM-BUILT UAVS

We built two custom UAVs for two main inspection purposes (see Fig. 2). The first UAV is a full-scale UAV designed for large-scale inspections. The UAV is based on a Gryphon Dynamics XV-1400 frame; it is equipped with a Nvidia TX1 GPU and an Auvidea j140 carrier. Three cameras are mounted on the UAV: a Sony DSC-QX30U, a FLIR with USB frame grabber, and an USB FPV 2mpix camera. The UAV uses Kongsberg Seatex MBR-144 OEM for radio communication. The UAV is powered by four Tattu 22000mAh 22.2V 25C 6S1P Lipo Battery packs which allow it to fly up to 42 minutes. The UAV can fly at an average speed of 60km/h

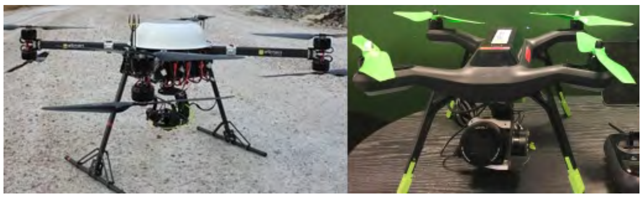
FIGURE 2. **Our custom-built full-scale UAV (left) for large-scale inspections and back pack UAV (right) for small-scale inspections.**

and can lift up to 40kg. The UAV is quite big and heavy; however, it is very stable when flying.

The second UAV is a backpack UAV built for small-scale inspections. The UAV is based on a 3DR Solo which is powered by a 5200 mAh 14.8V DC Lithium Polymer battery. It is customized by adding a custom gimbal from HDAir Studio and a Raspberry Pi computer for managing cameras. The UAV is equipped with a Sony DSC-QX30U camera. The UAV can fly up to 20 minutes at an average speed of 12km/h.

B. ACQUIRED OPTICAL IMAGES

Optical images are used as the main data source for inspection. Images are collected directly using cameras mounted on the UAVs. The UAVs are flown along power lines and circled around power masts to take pictures of the masts from different angles. For each power mast, around 20 images at 6048x4032 resolution are collected. The images are uploaded to the Microsoft Azure cloud after the flight for inspection.

C. DL-BASED DATA ANALYSIS AND INSPECTION

All images of power masts are analyzed by our component detection models to detect common power line components: poles, cross arms, top caps, and insulators. The detected components are then classified into more fine-grained power components classes using our component classification models and used as inputs for identifying faults. Images with potential faults will be assigned a higher priority in the inspection queue with the aim of reducing inspection time.

IV. DL VISION-BASED UAV INSPECTION

A. DATA ACQUISITION

Deep learning models for vision-based tasks typically require a huge amount of annotated data to train well. Unfortunately, to the best of our knowledge, there are no publicly available datasets that are big enough for satisfactory training of such models.

To move forward DL vision-based UAV inspection, we created four medium-sized datasets for training component detection and component classification models. The images in our datasets were collected using high quality DSLR cameras (e.g., Nikon D810, Canon EOS 5D Mark III, Nikon D3X) from helicopters and with multiple resolutions (e.g., 7360x4912, 6048x4032, 5760x3840). To increase the diversity of the data, we combined images from multiple power

TABLE 1. Properties of the DS1_Co, DS2_Tc, DS3_Po, and DS4_Cr datasets.

Dataset	Annotation type	#Images	Image size	Speed (images/hour)	Total time (hour)
DS1_Co	BB + label	28674	6048x4032	40	716.85
DS2_Tc	label	34002	256x256	1000	34
DS3_Po	label	26446	256x256	1000	26.5
DS4_Cr	label	34029	256x256	1000	34

grids in Norway, which were provided by Hafslund Nett and Troms Kraft.

The first dataset (DS1_Co), which is used for training component detection models, is annotated with bounding boxes (BB). The description of bounding boxes is (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) are the $(left, top)$ and $(right, bottom)$ locations of the bounding boxes. Each bounding box is associated with one of the 54 most common power line component classes that we selected. The selected component classes include three power pole classes (wooden poles, concrete poles, cracked poles), four cross arm classes (wooden cross arms, concrete cross arms, cracked cross arms, metal cross arms), three top cap classes (metal top caps, plastic top caps, missing top caps), a class for transformers, and 43 insulator classes including pin insulators, suspension insulators, and strain insulators.

The second dataset (DS2_Tc), which is used for training missing top cap detectors, is created by cropping top caps from images in the first dataset and annotating them with two classes, missing top caps and normal top caps. The third dataset (DS3_Po), which is used for training cracks in poles and woodpecker damage on poles detectors, is created by cropping poles from images in the first dataset, dividing the crops into overlapping squares, and annotating the squares with three classes, normal poles, cracked poles, and woodpecker-damaged poles.

The final dataset (DS4_Cr), which is used for training cracks on cross arms and rot damage on cross arms detectors, is created by rotating cross arm bounding boxes from images in the first dataset to remove background, cropping the rotated bounding boxes, dividing the crops into overlapping squares, and annotating the squares with four classes, cracked cross arms, rot-damaged cross arms, normal cross arms, and background. Properties and sample images of the four datasets are shown in Table 1 and Fig. 3.

As can be seen from Table 1, annotating images with labels only for training component classification models is quite fast; however, annotating images with both bounding boxes and labels for training component detection models is quite slow. The average annotating speed (in our experience) is 40 images/hour; thus, it requires around 717 hours to create our medium-sized DS1_Co dataset.

As discussed in our previous paper [1], these datasets come with many challenges. In the next sections, we describe the use of data augmentation techniques and our proposed multi-state component detection and classification approach to address the first three challenges of DL vision-based UAV

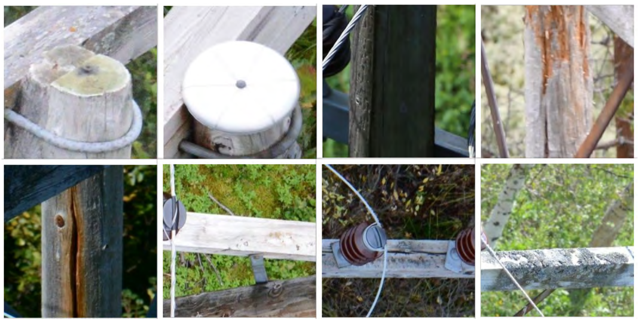
FIGURE 3. Sample images of the DS2_Tc, DS3_Po, and DS4_Cr datasets (from left to right, top to bottom): missing top cap, normal top cap, normal pole, woodpecker-damaged pole, cracked pole, normal cross arm, cracked cross arm, and rot-damaged cross arm.

inspection, including the lack of training data, class imbalance, and the detection of small components and faults.

B. DATA AUGMENTATION

Inspired by the successes of traditional data augmentation techniques in addressing the class imbalance and the lack of training data challenge [18], we propose a series of effective data augmentation techniques to generate more training data by applying transformations in the data-space.

To train a robust component detector for our pipeline, we combine original images with mast crops generated by the mast detector (see Fig. 5). When a mast is detected, its predicted bounding box is padded to be a square and cropped from the original image to generate one additional training image. The square is also slightly shifted in four directions (left, right, top, bottom) to generate four more training images. In addition, the training images are randomly flipped during training. These data augmentation techniques allow us to generate a training set that is 12 times bigger than our original training set.

To balance out the imbalanced classes and generate enough data to properly train our component classifiers, a series of effective data augmentation techniques are applied. All the data augmentation techniques are implemented using the scikit-image library [19].

The first technique involves adding Gaussian-distributed additive noise to account for noise that arises during image acquisition (e.g., sensor noise caused by poor illumination and/or high temperature, and/or transmission) [20]. The augmented image $f(i, j)$ is the sum of the true image $s(i, j)$ and the noise $n(i, j)$:

$$f(i, j) = s(i, j) + n(i, j). \tag{5}$$

The noise term, $n(i, j)$, follows a Gaussian random distribution:

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}, \tag{6}$$

where z represents the grey level, μ is the mean value, and σ is the standard deviation.

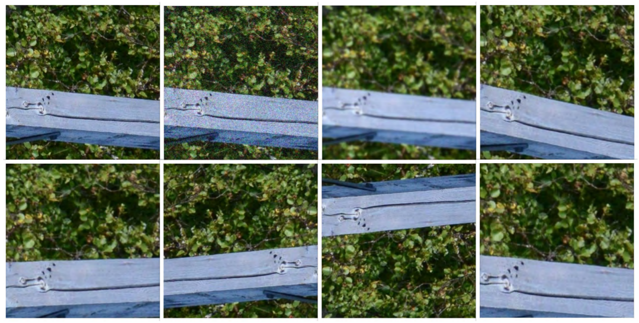
FIGURE 4. **Sample augmented images (from left to right, top to bottom): original image, image with Gaussian-distributed additive noise, Gaussian blurred image, left-rotated image, right-rotated image, horizontally flipped image, vertically flipped image, and center-cropped and zoomed in image.**

To account for possible out of focus, Gaussian blur is employed by convolving the image with a two-dimensional Gaussian function:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{7}$$

where x and y are distances from the origin in the horizontal axis and the vertical axis respectively, and σ is the standard deviation [21].

To account for various camera distances and viewing angles, zoom and rotation operators are performed [22]. The zoom operator is applied by randomly cropping images and scaling them to their original size. The rotation operator is performed by using a rotation matrix R:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix},$$

where θ is the rotation angle. The final technique involves flipping the images horizontally and vertically (see Fig. 4).

C. MULTI-STAGE COMPONENT DETECTION AND CLASSIFICATION

To tackle the detection of small power components and small faults challenge, we propose a multi-stage component detection and classification pipeline. The pipeline consists of five components connected as shown in Fig. 5. The pipeline works as follows: First, the mast detector detects power masts from input images. Then, the detected masts are cropped from the input images and used as inputs for the component detector to detect power components including top caps, poles, cross arms, and insulators. Finally, the detected top caps, poles, and cross arms are cropped from the input images and passed through their corresponding classifiers to identify faults. Algorithm 1 explains the workflow of the pipeline in detail. The pipeline allows us to mimic the "zoom-in" operation during inspection which enables the detection of small faults on power line components, such as cracks on poles and cross arms, woodpecker damage on poles, and rot damage on cross arms.

Algorithm 1 Multi-stage detection and classification

```
Input: Input image $I$, Mast detector's confidence threshold $m_{thres}$, Component detector's confidence threshold $c_{thres}$, Classification confidence threshold $cls_{thres}$, Mast detector $MD(I)$ that outputs bounding box coordinates ($m\_coords$) and confidence scores ($m\_confs$) of the detected masts, Component detector $CD(I)$ that outputs labels ($c\_labels$), bounding box coordinates ($c\_coords$), and confidence scores ($c\_confs$) of the detected components, Classifier name list $N$, Classifier list indexed by name $C$ (each classifier $CLF$ in C takes an image as input and outputs a label ($cls\_label$) and a confidence score $cls\_conf$)
Output: A list of detected and classified components $O$ (each item in $O$ contains a label, bounding box coordinates, and a confidence score)
  $m\_coords, m\_confs \leftarrow MD(I)$
  $m\_index \leftarrow argmax_i(m\_confs[i])$
  if $m\_confs[m\_index] > m_{thres}$ then
    $mast \leftarrow crop\_image(I, m\_coords[m\_index])$
  else
    $mast \leftarrow I$
  end if
  $c\_labels, c\_coords, c\_confs \leftarrow CD(mast)$
  for $c\_index \leftarrow 1, size(c\_labels)$ do
    $c\_label \leftarrow c\_labels[c\_index]$
    $c\_coord \leftarrow c\_coords[c\_index]$
    $c\_conf \leftarrow c\_confs[c\_index]$
    if $c\_conf > c_{thres}$ then
      $comp \leftarrow crop\_image(mast, c\_coord)$
      if $c\_label \in N$ then
        $CLF \leftarrow C[c\_label]$
        $cls\_label, cls\_conf \leftarrow CLF.classify(comp)$
      else
        $cls\_conf \leftarrow 0$
        $cls\_label \leftarrow NONE$
      end if
      if $cls\_conf > cls_{thres}$ then
        append [$cls\_label, c\_coord, cls\_conf$] to $O$
      else
        append [$c\_label, c\_coord, c\_conf$] to $O$
      end if
    end if
  end for
```

To select an object detector for implementing the mast detector and the component detector, we evaluate four state-of-the-art object detectors, which are SSD [2], YOLO [17], Faster R-CNN [23], and R-FCN [16], in terms of speed and mean Average Precision (*mAP*) [24]:

$$mAR = \frac{1}{|classes|} \sum_{c \in classes} AP(c), \tag{8}$$

where $AP(c)$ is the average precision of class c which takes the mean precision at a set of eleven equally spaced recall

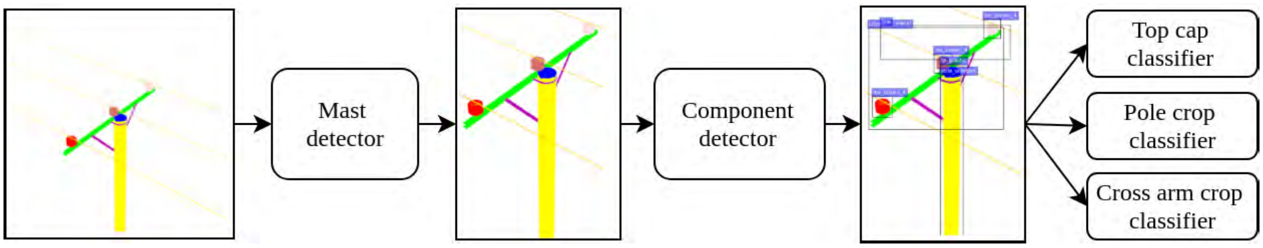

FIGURE 5. **The general structure of our proposed multi-stage component detection and classification pipeline. The pipeline consists of five components: a mast detector, a component detector, a top cap classifier, a pole crop classifier, and a cross arm crop classifier.**

levels [0, 0.1, ..., 1.0]:

$$\text{AP(c)} = \frac{1}{11} \sum_{r \in (0,0.1,\ldots,1.0)} p_{interp}(r), \tag{9}$$

where $p_{interp}(r)$ is an interpolated precision that takes the maximum precision over all recalls that exceed r:

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}), \tag{10}$$

where p and r are the precision and recall of class c:

$$p = \frac{TP(c)}{TP(c) + FP(c)}, \tag{11}$$

$$r = \frac{TP(c)}{TP(c) + FN(c)}, \tag{12}$$

where $TP(c)$, $FP(c)$, and $FN(c)$ are number of True Positives (TP), False Positives (FP), and False Negatives (FN) of class c respectively. A prediction is considered a true positive if its Intersection over Union (IoU) is greater than a predefined-threshold t (e.g., $t = 0.5$), otherwise it is considered a false positive. The IoU is defined as

$$IoU(G, P) = \frac{G \cap P}{G \cup P}, \tag{13}$$

where G and P are the ground-truth and the predicted bounding boxes respectively.

The performance of the four object detectors on the 21 most common insulator classes in our dataset is shown in Fig. 6 and Table 2. Although the SSD detector performs slightly worse than the R-FCN detector in terms of mAP, it is selected as our main object detector because of its speed, which is 3.47 times faster than that of the R-FCN detector.

TABLE 2. **Performance of the SSD, YOLO, Faster R-CNN, and R-FCN detectors on our dataset.**

Model	mAP	Speed (FPS)
SSD	**0.67**	**59**
Faster R-CNN	0.58	7
YOLO	0.61	45
R-FCN	0.68	17

In the last few years, many advanced CNN architectures have been proposed for image classification such as ResNet [3], Inception-v4 [25], and DenseNet [26]; however, ResNet was selected as our main classifier because it is easy to train, fast, and accurate.

V. EXPERIMENTS AND RESULTS

A. TRAINING

The component detectors and component classifiers are implemented using the Caffe [27] deep learning framework. To build the component detectors, we fine-tune the SSD512 model, which is pre-trained on the ILSVRC CLS-LOC dataset [24], using the Stochastic Gradient Descent optimizer (often shortened to SGD) with initial learning rate 0.001, 0.9 momentum, 0.0005 weight decay, and batch size 32 on four GPUs (3 Titan X Pascals and 1 GeForce GTX 1080 Ti).

The component classifiers are built by fine-tuning the ResNet50_cvgj [28] model, which is pre-trained on the ImageNet dataset [24], using the Adam optimizer [29] with initial learning rate 0.0001, 0.9 momentum1, 0.999 momentum2, 0.0001 weight decade, and batch size 16 on four GPUs (3 Titan X Pascals and 1 GeForce GTX 1080 Ti).

B. EXPERIMENTS

To evaluate the performance of our proposed multi-stage component detection pipeline and the effectiveness of our proposed data augmentation techniques, we conducted two experiments: a pipeline test and a data augmentation test. In the pipeline test, we compare against our proposed multi-stage component detection pipeline with data augmentation (MSCDP-Dataaug) and without data augmentation (MSCDP-Noaug) and a simple component detection model (SCDM) trained directly on original images. In the data augmentation test, we compare between the ResNet50_cvgj model trained with and without augmented data generated by our proposed data augmentation techniques for two tasks: pole crop classification and cross arm crop classification, respectively.

Since there are no publicly available datasets for power line inspection, both experiments are conducted on the DS1_Co, DS2_Tc, DS3_Po, and DS4_Cr datasets. The component detection models in all three methods in the first experiment are fine-tuned to detect ten component classes including poles, cross arms, top caps, and seven insulators classes on 80% of the images from the DS1_Co dataset. The remaining 20% of the images are used for evaluation. The models in the second experiment are trained on 80% of the data from

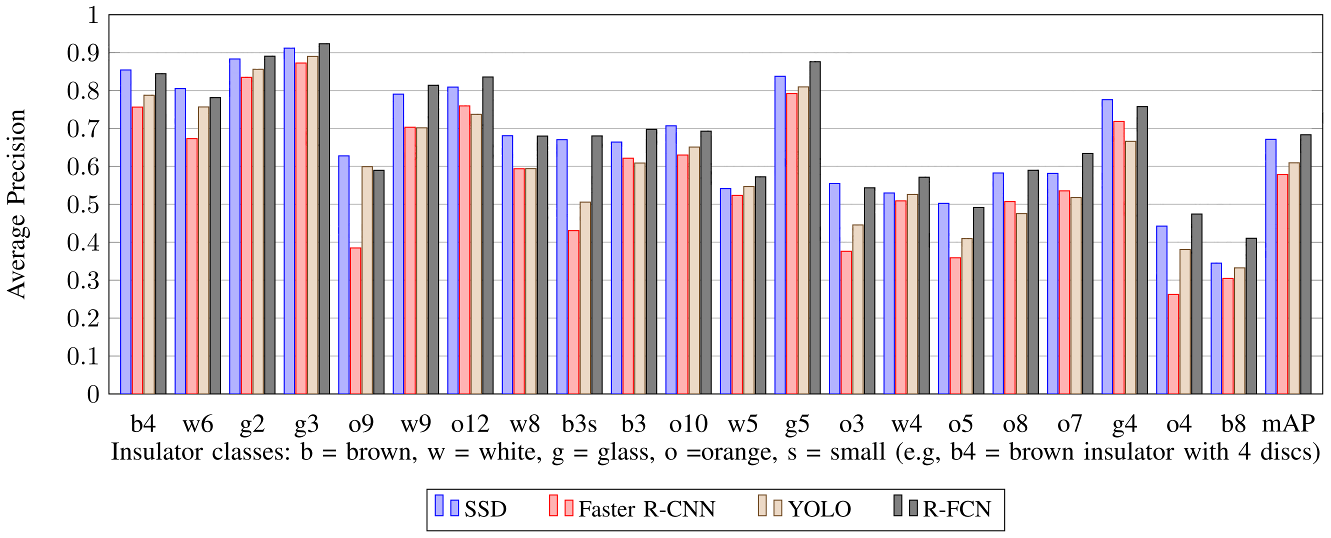

FIGURE 6. Mean Average Precision (mAP) of the SSD, YOLO, Faster R-CNN, and R-FCN detectors on the 21 most common insulator classes in our dataset.

the DS3_Po and DS4_Cr datasets. The remaining 20% of the data from the two datasets are used for testing.

The component detection models are evaluated in terms of *mAP* as defined in Equation 8. To account for class imbalance, the component classification models are evaluated in terms of weighted Precision (wP), weighted Recall (wR), and weighted F_1 score (wF_1):

$$wP = \frac{\sum_{c=1}^{|C|} sup(c) \cdot p(c)}{\sum_{c=1}^{|C|} sup(c)} \tag{14}$$

$$wR = \frac{\sum_{c=1}^{|C|} sup(c) \cdot r(c)}{\sum_{c=1}^{|C|} sup(c)} \tag{15}$$

$$wF_1 = \frac{\sum_{c=1}^{|C|} sup(c) \cdot F_1(c)}{\sum_{c=1}^{|C|} sup(c)} \tag{16}$$

where $sup(c)$, $p(c)$, $r(c)$, and $F_1(c)$ are the support, precision, recall, and F_1 score of class c respectively:

$$p(c) = \frac{TP(c)}{TP(c) + FP(c)} \tag{17}$$

$$r(c) = \frac{TP(c)}{TP(c) + FN(c)} \tag{18}$$

$$F_1(c) = 2 \cdot \frac{p(c) \cdot r(c)}{p(c) + r(c)} \tag{19}$$

where $TP(c)$, $FP(c)$, and $FN(c)$ are number of True Positives (TP), False Positives (FP), and False Negatives (FN) of class c respectively.

C. RESULTS

The detection results of the three methods in the pipeline test are shown in Table 3. Our proposed multi-stage component detection pipeline together with our proposed data augmentation techniques, i.e. the MSCDP-Dataaug method, achieves the best results in terms of *mAP* with 81.3% and outperforms the other two methods in 7/10 classes.

The test results of the pole crop classification and cross arm crop classification tasks in the data augmentation test are shown in Table 4 and Table 5 respectively. In both tasks, our proposed data augmentation techniques significantly improve wP, wR , and wF_1 score of the models. In particular, utilizing augmented data in training improves wP, wR , and wF_1 score by 8.93%, 8.42%, and 8.7% respectively on the pole crop classification task and by 1.98%, 0.56%, and 2% respectively on the cross arm crop classification task.

The top cap classifier is evaluated on a separate test set which consists of 681 missing top caps and 1103 normal top caps. The classifier achieves 0.987 weighted precision, 0.981 weighted recall, and 0.984 weighted F_1 score. Since the model achieves relatively high weighted precision, weighted recall, and weighted F_1 score, we skip data augmentation for this task.

VI. DISCUSSION

The testing results of the component detection models shown in Table 3 indicate that our proposed multi-stage component detection pipeline together with our proposed data augmentation techniques, i.e. the MSCDP-Dataaug method, can address the detection of small components and faults challenge. In particular, the MSCDP-Dataaug method achieves higher average precision on small insulator classes, such as insg3, inso9, insw9, insw6, and inso, compared to the simple component detection model, i.e. the SCDM method. In addition, the MSCDP-Dataaug method achieves 1.2% *mAP* higher than the SCDM method. This is due to the "zoom-in" operation enabled by our multi-stage component detection pipeline. By using outputs from the mast detector as inputs for the component detector, most of the irrelevant background is removed, and the relative sizes of the power components, especially the small ones, such as insulators and top caps, are significantly increased, resulting in richer features for the deep learning models to learn from.

TABLE 3. SCDM, MSCDP-Dataaug, and MSCDP-Noaug detection results on our medium-sized DS1_Co dataset.

Method	mAP	pole	cross arm	top cap	insb4	insg2	insg3	inso9	insw9	insw6	inso
SCDM	80.1	86.3	**85.7**	85.1	**84.2**	**90.6**	89.0	55.2	78.4	75.5	71.5
MSCDP-Noaug	78.5	85.9	83.4	84.3	78.7	87.4	87.4	53.8	78.9	76.5	69.1
MSCDP-Dataaug	**81.3**	**88.0**	83.4	**86.3**	82.4	89.0	**89.9**	**59.5**	**81.5**	**78.8**	**73.8**

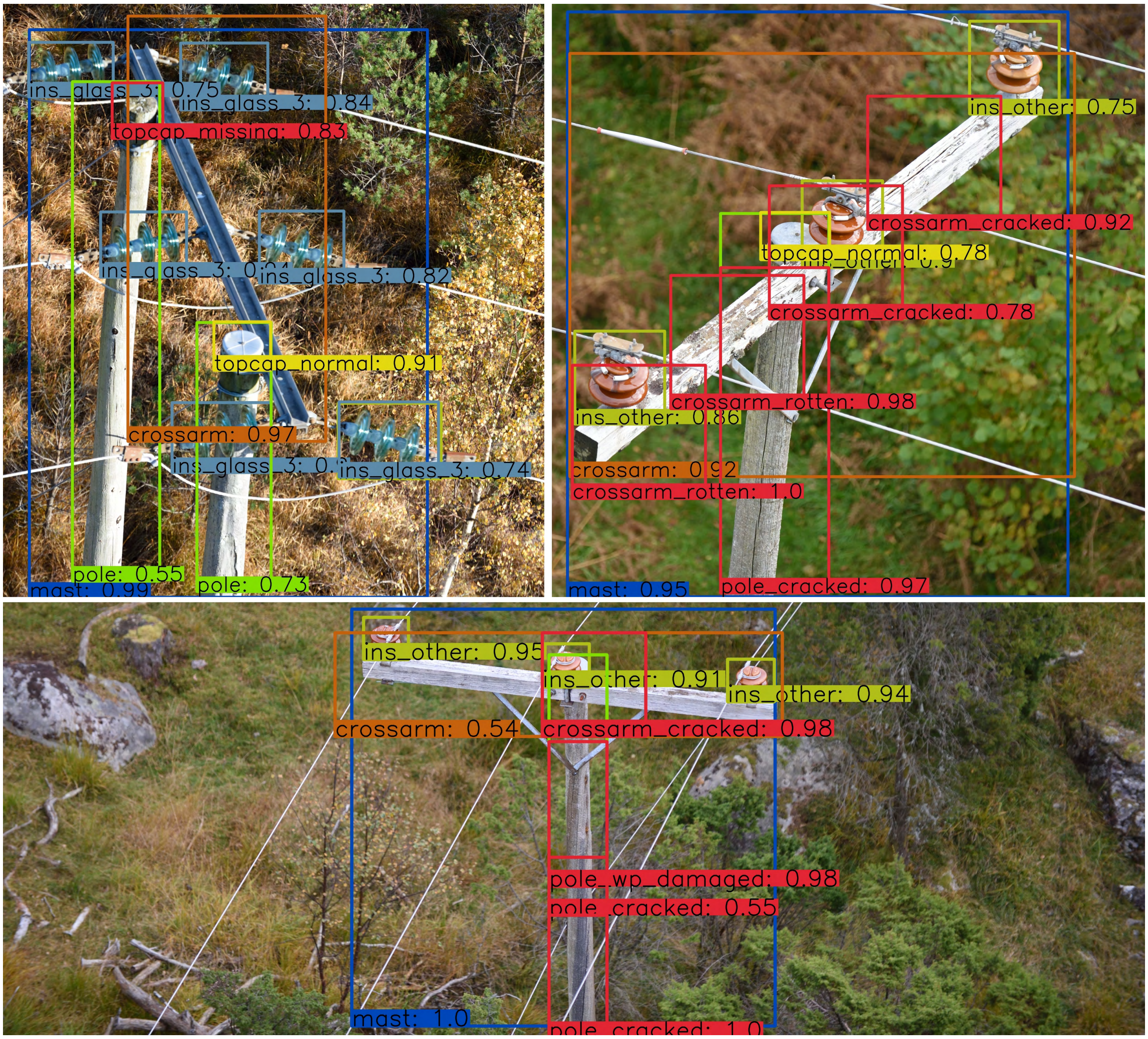

FIGURE 7. An illustration of power line component inspection using our proposed multi-stage component detection/classification pipeline. The pipeline is capable of detecting common power line components (e.g., insulators, poles, cross arms, top caps) and faults including missing top caps, cracks in poles and cross arms, woodpecker damage on poles, and rot damage on cross arms. In these three example images, our pipeline detected five faults (marked in red color): missing top cap (topcap_missing), rot damage on cross arm (crossarm_rotten), cracks in pole (pole_cracked), cracks in cross arm (crossarm_cracked), and woodpecker damage on poles (pole_wp_damaged).

The results shown in Table 3 also reveal that our proposed data augmentation techniques, especially the use of mast crops in training, can overcome the lack of training data challenge and significantly improve the performance of our proposed multi-stage component detection pipeline. In particular, our proposed pipeline without data augmentation, i.e. the MSCDP-Noaug method, performs worse than the SCDM method; the reason is that our proposed pipeline is

TABLE 4. **Pole crop classifier test results on the DS3_Po dataset.**

Method	wP	wR	wF_1 score
ResNet50_cvgj	88.00	88.67	88.31
ResNet50_cvgj+Dataaug	**96.93**	**97.09**	**97.01**

TABLE 5. **Cross arm cop classifier results on the DS4_Cr dataset.**

Method	wP	wR	wF_1 score
ResNet50_cvgj	72.02	83.54	75.96
ResNet50_cvgj+Dataaug	**74.00**	**84.10**	**77.96**

comprised of two models which typically require more training data compared to a single model in the SCDM method. However, our proposed pipeline with data augmentation, i.e. the MSCDP-Dataaug method, significantly outperforms the MSCDP-Noaug and the SCDM methods and improves *mAP* by 2.8% and 1.2% respectively. This suggests that data augmentation is crucial when addressing the lack of training data challenge for the component detection task.

The test results of the component classification models shown in Table 4 and Table 5 reveal that our proposed data augmentation techniques can address the class imbalance and the lack of training data challenge to some extent. In particular, using augmented data to balance out the imbalanced classes and increase the size of the training datasets improves wF_1 score in the pole crop classification and cross arm crop classification tasks by 8.7% and 2% respectively. In addition, models trained with augmented data in both tasks achieve higher *wP* and *wR* compared to models trained with original images only. Furthermore, as can be seen from Table 1, Table 4, and Table 5, wF_1 score of tasks with less training examples is improved more significantly when trained with augmented data. Specifically, while the cross arm crop classification task with 34029 training examples received 2% wF_1 score improvement when trained with augmented data, the pole crop classification task with only 26446 training examples, received much higher wF_1 score improvement of 8.7%. These results indicate that data augmentation is crucial when addressing the class imbalance and the lack of training data challenge for the component classification task.

The approaches proposed in this paper address the first three challenges of DL vision-based UAV inspection including the lack of training data, class imbalance, and the detection of small components and faults and facilitate the implementation of the automatic autonomous vision-base power line inspection system. The current version of our system is capable of detecting common power line components, such as poles, cross arms, top caps, insulators, and inspecting common faults on power line components including missing top caps, cracked poles, woodpecker-damaged poles, cracked cross arms, and rot-damaged cross arms (see Fig. 7).

When deployed in the Microsoft Azure cloud, with autoscale functionality and access to GPU VMs, our system has demonstrated its ability to analyze over 180,000 images per hour. This allows power utilities to inspect their power grids more often and at a lower cost than traditional inspection methods. Our system gives energy companies a fast and efficient tool to view the status of their infrastructure as well as export reports as a basis for their maintenance tasks. In addition, with the ability to access hard-to-reach areas and fly at high speed, our UAVs allow almost immediate assessment of power line damage after natural disasters for energy companies to plan for immediate repair or replacement work, which can greatly reduce the outage time and quickly reconnect the power grid.

Our system has, in our opinion, demonstrated its potential roles in the intelligent monitoring of power grids. Automatic autonomous UAVs equipped with sensors and cameras can automatically fly along power lines to perform online brief inspection to identify serious faults (e.g., collapsed poles, broken power lines, trees lying across and against power lines) and collect data for offline thorough inspection to detect potential faults that may lead to power outages such as broken insulators, missing top caps, cracked poles, woodpecker-damaged poles, cracked cross arms, and rot-damaged cross arms. These potential faults are very important information sources for electric utilities to make decisions for necessary repair or replacement work before any major damage that may cause power blackout.

VII. CONCLUSION AND FUTURE WORK

This paper presents a novel automatic autonomous vision-based power line inspection system that uses UAV inspection as the main inspection method, optical images as the primary data source, and deep learning as the backbone of the data analysis. To facilitate the implementation of the system, we address three main challenges of deep learning in vision-based power line inspection: (i) the lack of training data; (ii) class imbalance; and (iii) the detection of small power components and faults.

First, we create four medium-sized datasets for training component detection and classification models. Next, we propose a series of effective data augmentation techniques to generate more training data and balance out the imbalanced classes. Finally, we propose a multi-stage component detection and classification pipeline to detect small power components and faults.

The result indicates that the proposed approaches can address the three challenges and deliver significant improvement for detecting and classifying power line components. Compared with simple SSD detectors and ResNet50 classifiers, the proposed pipeline with data augmentation achieves 1.2% improvement in terms of *mAP* on the component detection task; using augmented data to balance out the imbalanced classes improves wF_1 score in the pole crop classification and cross arm crop classification tasks by 8.7% and 2% respectively. The proposed system can detect common faults on power line components: missing top caps, cracked poles, woodpecker-damaged poles, cracked cross arms, and rot-damaged cross arms at relatively high speed, over 18,000 images per hour.

With the aim of realizing a fully automatic autonomous vision-based power line inspection system, we propose two potential next steps: The first step is to combine the GPS way points-based, pole detection-based, and power line detection-based navigation approaches with UAV autopilots to facilitate self-driving UAVs. The second step involves upgrading the pipeline so that it can run directly on edge GPUs on UAVs to realize fully automatic autonomous online inspections.

ACKNOWLEDGMENT

The authors would like to thank eSmart Systems (Chi Hieu Huynh, Ngoc Hoang Tran, Dang Ha The Hien) and UiT Machine Learning Group (Michael Kampffmeyer) for support in the work with this paper.

REFERENCES

[1] V. N. Nguyen, R. Jenssen, and D. Roverso, "Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning," *Int. J. Elect. Power Energy Syst.*, vol. 99, pp. 107–120, Jul. 2018.

[2] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-46448-0_2

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[4] A. Savvaris, Y. Xie, K. Malandrakis, M. Lopez, and A. Tsourdos, "Development of a fuel cell hybrid-powered unmanned aerial vehicle," in *Proc. 24th Medit. Conf. Control Autom. (MED)*, Jun. 2016, pp. 1242–1247.

[5] T. Zhang *et al.*, "Current trends in the development of intelligent unmanned autonomous systems," *Frontiers Inf. Technol. Electron. Eng.*, vol. 18, no. 1, pp. 68–85, 2017.

[6] C. Deng, S. Wang, Z. Huang, Z. Tan, and J. Liu, "Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications," *J. Commun.*, vol. 9, no. 9, pp. 687–692, 2014.

[7] L. Matikainen *et al.*, "Remote sensing methods for power line corridor surveys," *ISPRS J. Photogramm. Remote Sens.*, vol. 119, pp. 10–31, Sep. 2016.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[11] J. Gu *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognit.*, vol. 77, pp. 354–377, May 2018.

[12] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proc. ICLR (Workshop Track)*, 2015, pp. 1–14.

[13] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[14] B. Xu, R. Huang, and M. Li. (2016). "Revise saturated activation functions." [Online]. Available: https://arxiv.org/abs/1602.05980

[15] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.

[16] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.

[18] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: When to warp?" in *Proc. Int. Conf. Digit. Image Comput., Techn. Appl. (DICTA)*, 2016, pp. 1–6.

[19] S. van der Walt *et al.*, "Scikit-image: Image processing in Python," *PeerJ*, vol. 2, p. e453, Jun. 2014, doi: 10.7717/peerj.453.

[20] A. K. Boyat and B. K. Joshi, "A review paper: Noise models in digital image processing," *Signal Image Process.*, vol. 6, no. 2, p. 63, 2015.

[21] L. Shapiro and G. Stockman, *Computer Vision*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2001, chs. 5.4, p. 154.

[22] G. Wolberg, *Digital Image Warping*, vol. 10662, Los Alamitos, CA, USA: IEEE Computer Society Press, 1990, chs. 3.3, pp. 47–49.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[24] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[25] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI*, vol. 4, 2017, p. 12.

[26] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4700–4708.

[27] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[28] M. Simon, E. Rodner, and J. Denzler. (2016). "ImageNet pre-trained models with batch normalization." [Online]. Available: https://arxiv.org/abs/1612.01452

[29] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

VAN NHAN NGUYEN received the B.Eng. degree in computer science and engineering from the Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, in 2014, and the M.S. degree in computer science from Østfold University College, Halden, Norway, in 2016. He is currently pursuing the Ph.D. degree in deep learning with the UiT Machine Learning Group, UiT/The Arctic University of Norway, Tromsø, Norway. His research interests include deep vision (deep learning for computer vision), especially image classification, object detection, semantic segmentation, one-shot learning, zero-shot learning, deep learning-based line detection, and vision-based automatic autonomous inspection of power lines based on unmanned aerial vehicles (UAV) and deep learning.

ROBERT JENSSEN (M'02) received the Ph.D. (Dr. Scient.) degree in electrical engineering from the University of Tromsø, in 2005. He was a Guest Researcher with the Technical University of Denmark, Kongens Lyngby, Denmark, from 2012 to 2013, also with the Technical University of Berlin, Berlin, Germany, from 2008 to 2009, and also with the University of Florida, Gainesville, FL, USA, from 2002 to 2003, spring 2004, and spring 2018. He is currently a Professor with the Department of Physics and Technology, UiT/The Arctic University of Norway, Tromsø, Norway. He directs the UiT Machine Learning Group: http://site.uit.no/ml. He is also a Research Professor with the Norwegian Computing Center, Oslo, Norway. He is on the IEEE Technical Committee on Machine Learning for Signal Processing. He is the President of the Norwegian Section of IAPR (NOBIM - nobim.no) and serves on the IAPR Governing Board. He is an Associate Editor of the Journal *Pattern Recognition*, since 2010.

DAVIDE ROVERSO received the Ph.D. degree in computing science. He has over 25 years' of experience in the field of machine learning and Big Data Analytics, with the applications in diagnostics, prognostics, condition monitoring, and early fault detection in complex processes, in sectors ranging from energy to medicine and environmental monitoring. He has authored over 100 publications in international journals, conference proceedings, and edited books. He is currently the Chief Analytics Officer at eSmart Systems, where he leads the Analytics and Data Science Group.