Paper 19

Intelligent Damage Classification and Estimation in Power Distribution Poles Using Unmanned Aerial Vehicles and Convolutional Neural Networks

This paper develops a model to automate the process of estimating and localizing damages in power distribution poles, which utilizes the images taken by UAVs transferred in real-time to an intelligent damage classification and estimation (IDCE) unit. The IDCE unit integrates four convolutional neural networks to learn the states of poles from images, extract the image characteristics, and train an automated intelligent tool to replace manual fault location and damage estimation. The proposed model first determines the type of pole damages, including falling and burning, and then estimates the percentage of damage in each type. The IDCE unit also localizes damages in the poles by locating possible burning or arcing parts.

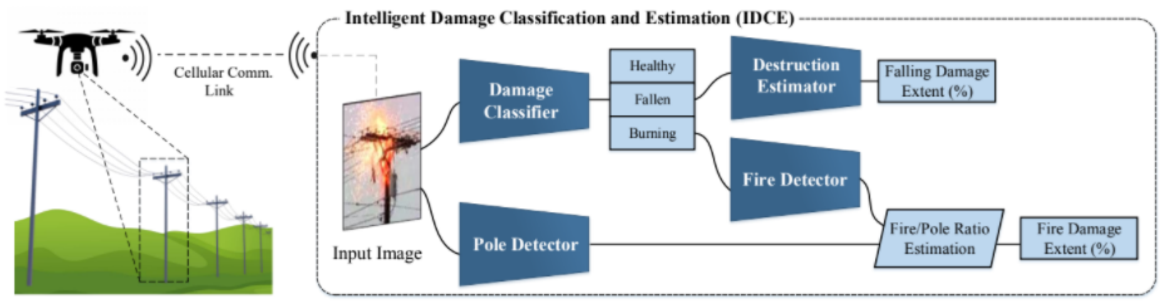
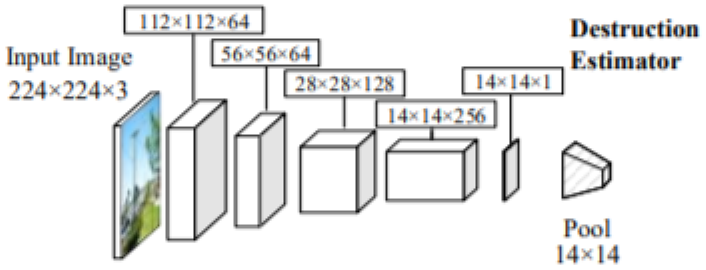
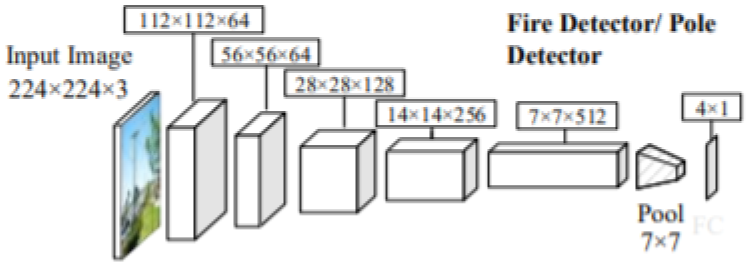


Fig. 1. Architecture of Intelligent Damage Classification and Estimation Unit

The proposed architecture, consisting of four CNNs, can efficiently capture nuances in images of different resolutions, scales and backgrounds and promises to reduce inspection and damage detection time. The ResNet architecture is used as a base for the CNNs, both for classification and regression purposes, creating a series of networks capable of classifying the damage and estimating its extent on each pole. A data set of 1615 images is utilized to train, validate and test the proposed model, which demonstrates high accuracy of the model in classifying and estimating damages in distribution poles.

The proposed model estimates the damage extent in fallen poles based on their apparent tilting or breaking, while in burning and arcing poles the damage is estimated by the ratio of the areas of fire to pole bounding boxes. Although high accuracy is achieved in the classification of healthy and fallen poles, burning and arcing poles proved harder to detect. The specific topic of training networks to detect fire in poles has a particular application in power systems and may be of interest in future research. In particular, image processing methods and larger training image dataset may contribute to higher accuracy of detecting burning poles.





Paper 20

Real-Time On-Board Deep Learning Fault Detection for Autonomous UAV Inspections.

In this paper proposed and evaluated the use of autonomous DL algorithms running in real time on-board UAVs with the purpose of visual power line inspection. Also compared the results between different SBDs that can be embedded in UAVs for running inference for powerful DL models. Compared real-time results among YOLOv3-tiny, YOLOv3-darknet53, YOLOv4-tiny, and YOLOv4 (CSPdarknet-53) and investigated the real-world impact of weight optimization using TensorRT. Model trained a YOLOv4-tiny architecture-based deep neural network, as it showed prominent results for detecting components with high accuracy. It realized that YOLOv4-tiny showed higher accuracy with a better frame rate during real-time inspection; YOLOv3-darknet53 also showed an acceptable level in terms of accuracy, but it showed lower FPS during detection.

Table 1. Accuracy and average loss results of different DL models on different iterations.

Evaluation $\longrightarrow$	**10,000 Iterations**		**20,000 Iterations**	
$\downarrow$ DL Models	**Accuracy**	**avg. Loss**	**Accuracy**	**avg. Loss**
YOLOv3-tiny	74%	1.08	78.3%	0.97
YOLOv4-tiny	81.2 %	0.23	84.4%	0.15
YOLOv3-full	79.9 %	0.29	84.3%	0.11
YOLOv4-full (Mish)	82.4 %	0.38	83.6%	0.44
YOLOv4-full (leaky)	81.6 %	0.35	82.8 %	0.21

For running such deep learning models in a real-time environment, different single-board devices such as the Raspberry Pi 4, Nvidia Jetson Nano, Nvidia Jetson TX2, and

Nvidia Jetson AGX Xavier were used for the experimental evaluation. Our experimental results demonstrated that the proposed approach can be effective and efficient for fully automatic real-time on-board visual power line inspection.

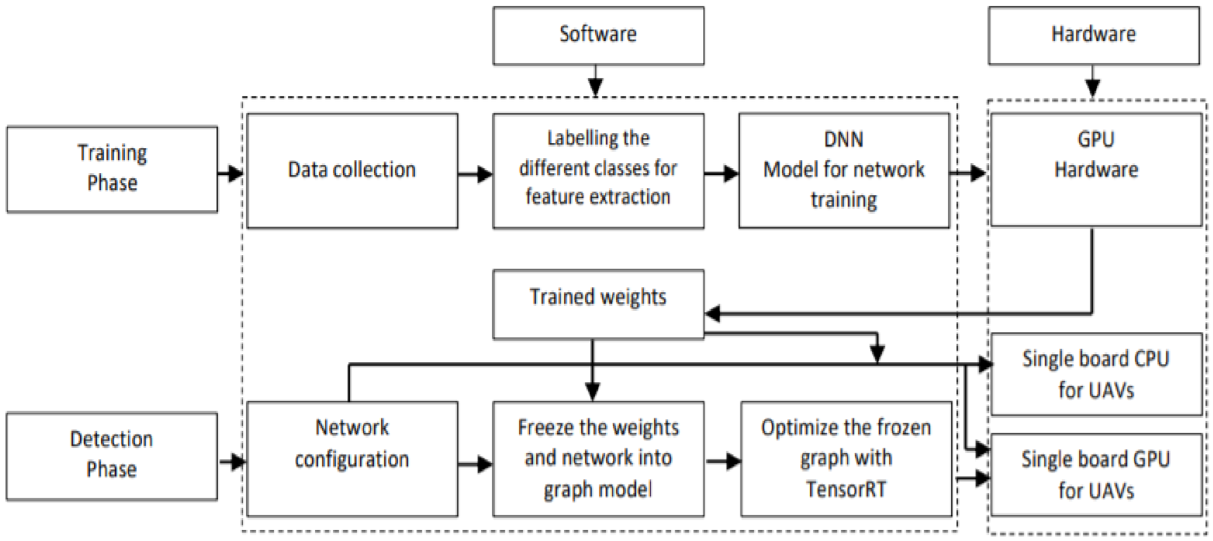

Software
Hardware
Training Phase
Data collection
Labelling the different classes for feature extraction
DNN Model for network training
GPU Hardware
Trained weights
Single board CPU for UAVs
Detection Phase
Network configuration
Freeze the weights and network into graph model
Optimize the frozen graph with TensorRT
Single board GPU for UAVs

Figure 3. Proposed architecture for UAVs based on-board inspection and an autonomous vision system (Adapted from ref. [4]).

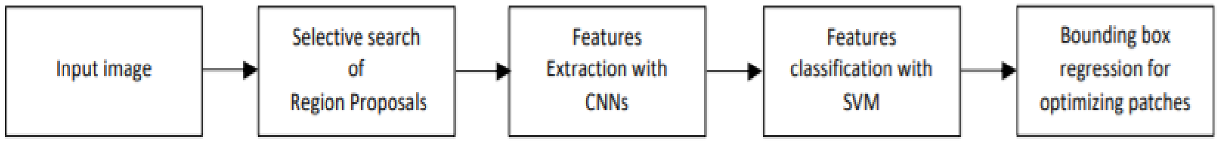

Input image
Selective search of Region Proposals
Features Extraction with CNNs
Features classification with SVM
Bounding box regression for optimizing patches

Figure 5. Flowchart of the R-CNN model (Adapted from ref. [4]).

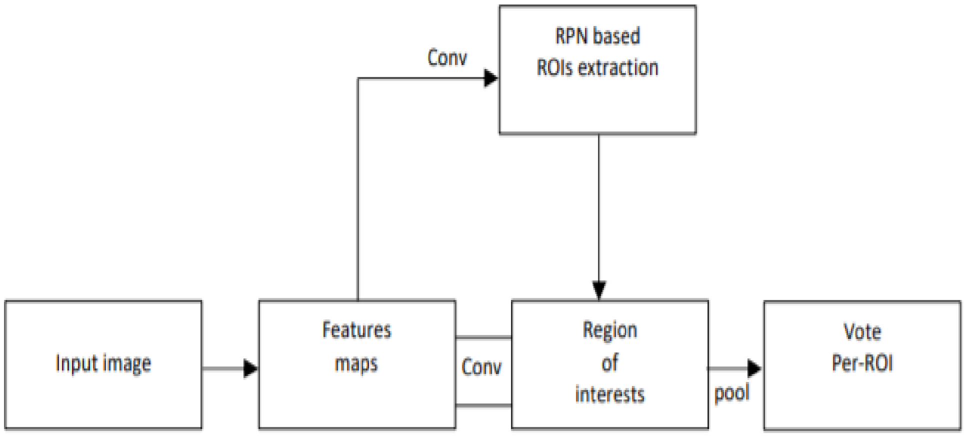


Figure 6. Region-based fully convolutional network model (Adapted from ref. [4]).

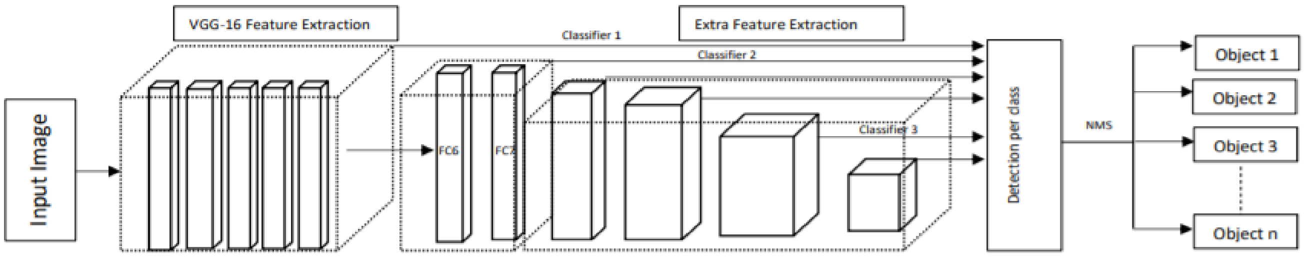


Figure 7. Architecture of the SSD model.

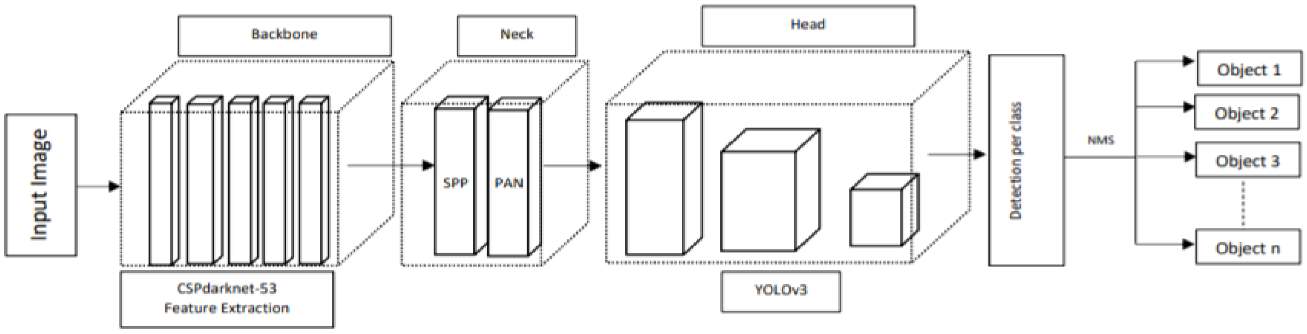


Figure 8. Architecture of the YOLOv4 model.