International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-4, February 2020

Training a Deep Learning Network with an Insignificantly Small Dataset

T. Kavitha, K. Lakshmi

Abstract: ***In the last few years, Deep Learning is one of the top research areas in academia as well as in industry. Every industry is now looking for a deep learning-based solution to the problems in hand. As a researcher, learning "Deep Learning" through practical experiments will be a very challenging task. Particularly, training a deep learning network with huge amount of training data will make it impractical to do this on a normal desktop computer or laptop. Even a small-scale application in computer vision using deep learning techniques will require several days of training the deep network model on a very higher end Graphical Processing Unit (GPU) clusters or Tensor Processing Unit (TPU) clusters that makes impractical to do that research on a conventional laptop.***

In this work, we address the possibilities of training a deep learning network with an insignificantly small dataset. Here we mean "significantly small dataset' as a dataset with only few images (<10) per class. Since we are going to design a prototype drone detection system which is a single class classification problem, we hereby try to train the deep learning network only with few drone images (2 images only).

Our research question is: will it be possible to train a YOLO deep learning network model only with two images and achieve a descent detection accurate on a constrained test dataset of drones? This paper addresses that issue and our results prove that it is possible to train a deep learning network only with two images and achieve good performance under constrained application environments.

Keywords: Computer Vision, Convolutional Neural Networks, Deep Learning, GPU, Object Detection, TPU, Unmanned Aerial Vehicles, YOLO.

I. INTRODUCTION

The recent advancements in the research of neural networks has its focus on computational power, emergence of new algorithms, and a significant improvement in the labelled data. The applications of recent research in neural network requires high computational requirements for network training which restricts its usage in many areas.

Neural networks play a major role in state-of-the-art research in computing field. Though it has high computational complexity, it is used in many areas due to its computational accuracy.

Revised Manuscript Received on February 28, 2020.

***** Correspondence Author

T.Kavitha*, Research Scholar, Department of CSE, Periyar Maniammai Institute of Science & Technology, Vallam, Thanjavur-613 403, India. Email: tkavitha07@yahoo.com

K.Lakshmi, Professor, Department of CSE, Periyar Maniammai Institute of Science & Technology, Vallam, Thanjavur-613 403, India. Email: lakshmi@pmu.edu

A. Difficulties in Experimenting and Doing Research on Deep Learning Deep learning is the top ranked technique among a few thousand techniques related with healthcare, materials, energy etc. Deep learning is becoming an eminent technique in image detection and classification, self-driving cars, quality control in production and maintenance and bio-medical imaging. Now-a-days the power of deep learning spreads in many applications such as drug design, cybersecurity, chip design, voice recognition, and optimizing manufacturing operations.

Correspondence Author

B. Issues of Deep Learning

Though the deep networks have its widespread use in several applications, the following issues must be addressed very well before using the deep learning.

- Availability and Processing of large datasets

Though we have plenty of data in real life, the availability of large datasets for training is still a big problem in most of the industrial and consumer applications. Once it is collected, we need to train the network with the available dataset. Training the network takes a longer time to process the large datasets in CPU, GPU, or TPU based systems. To overcome this challenge, the cutting-edge techniques such as transfer learning and adversarial networks are widely used.

- Difficulty in identifying the hyperparameters

Deep learning works well in many of the applications due to its structure such as the number of interconnected neurons and free parameters. These parameters capture subtle variations in the data. However, it is difficult to identify hyperparameters before training. This issue may be overcome by some of the methods such as manual search, grid search, random search, and Bayesian Optimization. Sometimes, overfitting of data occurs when huge number of parameters present.

- Difficulty in retraining the model

The edges are difficult to implement due to its size even with large deep learning networks. Since the deep learning networks require a lot of time for training, it is difficult to retrain the models quickly on the edges with the available new information.

- Difficulty in understanding the insight of deep networks

It is difficult to understand what the deep networks do internally and how it arrives the solution due to its significant number of layers and nodes and its interconnections. Understanding the insight of deep networks will be useful for decision making applications.

- Difficulty in maintaining stability of results

The deep networks may produce different results when a little amount of noise is added to the input data. It makes a susceptible and unstable network even for small changes in the input data. It paves a way for hackers.

C. Challenges of supervised learning [2]

It is difficult to understand the following concerns:

- The number of data required for the basic mapping function from input to output.
- The number of data required to estimate the performance of the mapping function approximation.
- The number of data required for training and testing to avoid underfitting and overfitting of data.

D. Hardware Requirement of Deep Learning

Since deep learning is computationally intensive, we need to put much concern on hardware requirements. The CPU that we use for tasks with very little computations are not suitable for extensive computations which require a lot of parameters. For a computationally intensive tasks, there are many cost based third-party services provide space to perform computations which is expensive. To overcome this hardware issue, purchasing of expensive Graphical Processing Unit (GPU) is highlighted.

Few of the hardware peripherals related with deep learning are seen below.

CPU: Central Processing Unit, or Processor is generally used which is capable of performing various operations with or without computations and memory transfers.

GPU: Graphical Processing Unit has huge processing power which grows in recent years due to its use in computer games and graphic engines. GPU performs parallel programming by GPUs and CPUs which processes and analyses the data just as an image or any kind of graphic forms. GPUs are well suited for both graphical processing and for scientific computing. GPUs are rapidly used in many applications that involve video and image processing such as rendering of video, image transformations, and in compression techniques etc.

Nvidia: It has parallel processing power and comes along with GPU which improves the processing speed of high-performance applications. Nvidia interacts with GPU with the help of a high-level language 'CUDA'. It can be able to perform any kind of graphical processing.

ASIC: It is an Application Specific Integrated Circuit which is a personalized chip to meet a specific functionality. It is a hardware component, no programming involved.

TPU: Tensor Processing Unit is a Google's product which is specifically designed chip to accelerate the neural computations. Both CPU and GPU work with matrices. It can handle huge number of additions and multiplications for neural networks. GPU has some constants in its memory whereas a TPU requires a constant flow of data.

We can avail TPU cloud services to access the TPU used in Google's datacenters which is not available in the market. Since it has a dedicated infrastructure, we can access only through TPU cloud services.

Coral edge TPU is available for public use which is a lesser capable device compared with the normal TPU cloud services.

E. Data Requirements for Deep Learning

When we use deep learning, the requirements for data are greater than other methods used for analytics [3].

To use neural networks effectively, we need a large collection of labelled data for training in addition to computing infrastructure. The deep learning techniques are particularly useful in pattern extraction from large, complex, and multidimensional data types such as video, audio, speech, and an image [3].

Deep-learning techniques require massive amounts of data records for an effective modeling to produce good classification results. An acceptable performance is attained by approximately with 5000 labeled samples per class and it may match or exceed the performance in human level with at least 10 million labelled training samples. If sufficient datasets are not available, it is possible to create an enhanced dataset [3].

These huge data sets can even be difficult to obtain or create for many businesses and assigning labels to them is a challenging task. Most of the supervised learning techniques require manual labelling and categorization. To overcome this problem, the techniques such as transfer learning, generative adversarial networks etc. exist. In these techniques, the model learns by themselves with the help of a large number of samples [3].

Based on the context, we have to collect and enhance the dataset at large scale for an effective training which will improve the performance. These large datasets may sometimes cause either overfitting or underfitting. In overfitting, a model matches the noisy or random features of the training set which results in a lack of accuracy. In underfitting, the model fails to capture all of the relevant features [3].

F. Previous Works on Learning from Limited Data

With the limited set of data, there exist a learning problem in different orientations. Firstly, a high degree of work involved in few-shot and one-shot learning. It is assumed that a given a set of classes with enough training will be used to improve the performance in the other set of classes with very few categorized samples. They learn and generalize only the highly discriminative features from the given dataset that suits well to the new classes. Hence, the classification with limited training data is performed well by the nearest neighbor approach. Secondly, meta learning approach is used for few-shot learning. It means that training is given on huge datasets for the learner to learn from small datasets [23].

Now-a-days, deep learning methods are popularly used when sufficient amount of data available. It is predominantly used not only by large datasets but also the huge developments in the hardware components. However, in most of the applications, there is still a problem in collecting training data due to expensive cost involved in collecting and annotating them. In such cases, pre-training on similar tasks is given with the exiting dataset [23].

This method is known as transfer learning as the training is given with pre-trained models without spending on data collection [23].

It leads to remarkable improvements in the research field.

But this method is problematic in two domains namely image analysis in medical field and in huge imagery datasets. In these two fields, the data are highly sensitive so that the transfer learning is not suitable for training

with pre-trained models. However, most of the deep learning-based research with training on small datasets use the transfer learning.

The term "small dataset" is highly subjective which depends on the domain we choose and the number of classes involved in the data [23].

II. MODELING

A. Components of the Learning Algorithm

A number of components and hyperparameters involve in the training of a deep networks. They are given as follows:

- Error Function

An error function is otherwise known as the objective function, cost function, or the loss function which has to be chosen. A specific set of values should be given to the parameters for maximum likelihood. Cross Entropy and Mean Squared Error are commonly used loss functions to deal with the classification and regression problems respectively.

- Loss Function

A function which estimates the performance of the model by a set of weights taken from training the samples.

For optimizing the process, it is necessary to give a starting point which contains initial model parameters or weights from which the model updates are done.

Initial starting point is carefully chosen for the optimization algorithm due to non-convex error surface. The initial weights given to the model are selected as small random values. There exist some different techniques known as weight initialization methods for scaling and distribution of these values.

- Weight Initialization

Small random values are initialized to model weights which are required for a model in the beginning of the training process.

The model error or loss is calculated from the examples of training dataset while updating the model. If the datasets are smaller, all the examples in the training dataset are used for calculating the model error. If the data is frequently changing, a single example is used. When we use hybrid approach, the number of examples is chosen from the training dataset. The error gradient is calculated based on the batch size which tells the number of taken samples.

- Batch Size

The error gradient is estimated by the number of examples prior to updating the parameters of the model.

After estimating the error gradient, the error derivative is calculated to update the parameters. There is a possibility for statistical noise in the examples of the training dataset and in estimating the error gradient. When the number of layers and parameters of the model are updated separately, it is difficult to calculate how much changes exactly to be done for each model parameter for reducing the error gradient.

Rather, a small portion of the weight update is performed in each iteration. The model weights and time taken to learn from the training dataset are controlled by the hyperparameter called learning rate.

- Learning Rate

The rate in which each model parameter is updated in every iteration of the learning algorithm to provide better performance.

An enough and good set of model parameters are found in the repeated training process of a model. The number of iterations of the process depends on the number of complete passes over the training dataset. Once it is over, the training process will be terminated.

- Epochs

The number of complete passes through the training dataset before the training process is terminated.

The above discussed hyperparameters control the learning algorithm of deep networks.

III. OBJECT DETECTION USING DEEP LEARNING NETWORK

There are several object detection algorithms with different capabilities. These algorithms are mostly split into two groups according to how they perform their tasks.

The first group is composed of algorithms based on classification and work in two stages. First, they select the interesting parts of the image, and then they classify objects within those regions using Convolutional Neural Networks (CNN). This group, which includes solutions such as R-CNN, is usually too slow to be applied in real-time situations.

The algorithms in the second group are based on regression and they scan the whole image and make predictions to localize, identify and classify objects within the image. Algorithms in this group, such as You Only Look Once (YOLO), are faster and can be used for real-time object detection.

Deep learning is one of the most exciting technologies of the last few years. Most of the well-known applications of AI such as Speech Synthesis, Image and Video Processing and Natural Language Processing are driven by Deep Learning. The deep learning algorithms resembles the human brain using artificial neural networks and gradually learn to solve a given problem accurately.

A. YOLO Object Detection

You Only Look Once (YOLO) is a network that uses Deep Learning (DL) algorithms for object detection. YOLO performs object detection by classifying certain objects within the image and determining where they are located on it.

For example, if you input an image of a herd of sheep into a YOLO network, it will generate an output of a vector of bounding boxes for each individual sheep and classify it as such.

How YOLO improves over previous object detection methods-

Previous object detection methods like Region-Convolutional Neural Networks (R-CNN), including other variations of it like fast R-CNN, performed object detection tasks in a pipeline of multi-step series. R-CNN focuses on a specific region within the image and trains each individual component separately.

This process requires the R-CNN to classify 2000 regions per image, which makes it very time-consuming (47 seconds per individual test image). Thus it, cannot be implemented in real-time. Additionally, R-CNN uses a fixed selective algorithm, which means no learning process occurs during this stage so the network might generate an inferior region proposal.

This makes object detection networks such as R-CNN harder to optimize and slower compared to YOLO. YOLO is much faster (45 frames per second) and easier to optimize than previous algorithms, as it is based on an algorithm that uses only one neural network to run all components of the task.

To gain a better understanding of what YOLO is, we first have to explore its architecture and algorithm.

B. YOLO Architecture

A YOLO network consists of three main parts. First, the algorithm, also known as the predictions vector. Second, the network. Third, the loss function.

- The YOLO Object Detection Algorithm

The computing process of YOLO is discussed as follows:

Step1: Split the input image into an S×S grid.

Step2: Predict B bounding boxes from each grid cell which encloses an object and the corresponding confidence scores to find out whether the predicted bounding box has an object. It is symbolized as $\boldsymbol{x, y, w, h, Pr(Object) \times IOU_{Pr}^{Gt}}$, where Pr*(Object)* is the probability that the current position is a valid object class, IOU is the overlap probability between the predicted(Pr) bounding box and the ground truth(Gt). The *x*, *y* is the center coordinate and the *w*, *h* is the size of the box.

Step 3: Calculate the conditional probability of prediction class in each grid $\boldsymbol{C_i = Pr(Class|Object)}$.

Step 4: Multiply both the conditional probability of a class and confidence while predicting the object as

$$\Pr(Class_i \mid Object) \times \Pr(Object) \times IOU_{\Pr}^{Gt} = \Pr(Class_i) \times IOU_{\Pr}^{Gt}$$

Non-maximally suppressed operations are inevitably performed in the presence of large objects. Since B has a value of 2, a grid will only return two boxes, and will have only one category. If there are multiple categories in a grid, there will be a problem. YOLO results good for images of small objects.

- The Network

A YOLO network is structured like a regular CNN, it contains convolutional and max-pooling layers and then two fully connected CNN layers.

- The Loss Function

We only want one of the bounding boxes to be responsible for the object within the image since the YOLO algorithm predicts multiple bounding boxes for each grid cell. To achieve this, we use the loss function to compute the loss for each true positive. To make the loss function more efficient, we need to select the bounding box with the highest Intersection over Union (IoU) with the ground truth. This method improves predictions by making specialized bounding boxes which improves the predictions for some aspect ratios and sizes.

B. YOLO Versions v3

The most current version of YOLO is the third iteration of the object detection network. The creators of YOLO designed new versions so to make improvements over previous versions, mostly focusing on improving the detection accuracy. YOLO v3 is an incremental upgrade over YOLO v2, which uses another variant of Darknet. This YOLO v3 architecture consists of 53 layers trained on Imagenet and another 53 tasked with object detection which amounts to 106 layers. While this has dramatically improved the accuracy of the network, it has also reduced the speed from 45 fps to 30 fps. Training a YOLO model for object detection requires running multiple trials which can be highly challenging in terms of running and tracking these experiments. These challenges can lead to excessive storage and hardware costs and time consumption.

- Darknet [17]

It is written in C and CUDA. It acts as a basis for YOLO (You Only Look Once) which is useful for modeling deep learning systems. Both the CPU based and GPU based computations are supported by Darknet. The source is provided by GitHub which can be easily used by the users for implementing the logic.

We can install Darknet with the help of only two optional dependencies: OpenCV and CUDA. It is very fast and easy to install and configure for training. If we want to use a wide variety of image types, it is supported by OpenCV whereas GPU computation is supported by CUDA. Neither is compulsory but the users can start working by just installing the base system. It has only been tested on Linux and Mac computers.

The Darknet framework explores state-of-the-art real-time object detection system called You Only Look Once (YOLO) . Darknet displays information as it loads the config file and weights then it classifies the image and prints the classes for the image. When the data changes over time, Recurrent Neural Networks are used as powerful models for representing data. Darknet can handle them without using CUDA or OpenCV.

In this work, we used Alexey's [13] implementation of Darknet which is a fork of Joseph Redmon's original Darknet [14] implementation. Since it is based on C, and it contains features to use YOLO, we opted to use it as the best choice for designing our fast, real-time drone detection system.

IV. RESULTS AND DISCUSSION

A. The Small Dataset used for Training

In this work, we only used the following two images to form a tiny "2 image dataset" from [15] and used it for training. Along with the image, the ground truth bounding box information is available.

Training Image 1	Training Image 2
	

Figure 1. The Dataset with only Two Images

The deep learning network will be trained with this data by repeatedly introducing the above two images as training images.

B. About the Batch Size and Epochs

As far as these two image small dataset is concerned, the batch size, subdivisions size and epochs are too important for successful training.

To make the training as a successful and progressive process, we used the batch size of 2 and subdivisions size of 1. It means that we are training both the images in each epoch and repeated it for 700 epochs.

C. Metrics Used for Evaluation

We used the following metrics for evaluating the drone detection system.

D. Mean Average Precision (mAP)

AP (Average precision) is a popular metric in measuring the accuracy of object detectors. Average precision computes the average precision value for recall value over 0 to 1

Average Precision (AP) is averaged over categories and it is called "mean average precision" (mAP).

In this work, we used mAP at IoU of 0.5 ($mAP^{IoU=0.5}$)

The following table shows the performance improvement in training with the same training image 2 at a minimum detection threshold of 0.1

Table 1. Performance Improvement over Epochs

Epoch of Training	Detected Bounding Boxes of Potential Drone Region	Description
100		With insufficient training, several bounding boxes were selected at threshold 0.1
200		The Drone was detected at 0.88% Confidence
300		The Drone was detected at 16.24% Confidence
400		The Drone was detected at 21.52% Confidence

To see the obvious improvement in performance at initial stage of training, we plotted three values that are showing progressive improvement with respect to confidence.

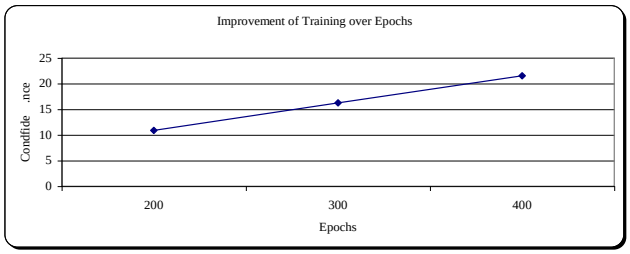

Figure 2. Increasing Performance over Epochs

E. Performance of Detection with Random Test Images

To test the performance of a trained network for the capability of detecting drones from completely different images, we chose some of the random images from[15] and from the Internet and showed successful detection of drones in them.

The following result shows the capability of the trained network to detect a drone from a terrain of different kinds of areas (sky, water, and land).

Figure 3. Drone Detected on a Test Image (700 epochs Training)

The following result shows the capability of the trained network to detect a drone in a cloudy background.

Figure 4. Drone Detected on a Test Image at 800 epochs of Training

The following result show the capability of the trained network to detect multiple drones in a scene. Of course, the detection system missed some of the partially exposing drones in the image. The reason is that we only trained the network with two images of drones.

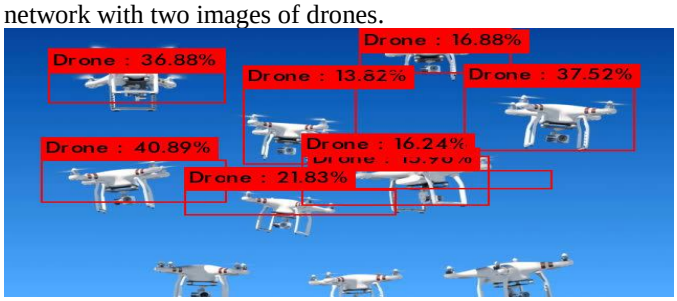

Figure 5. Multiple Drones Detected on a Test Image at 800 Epochs of Training

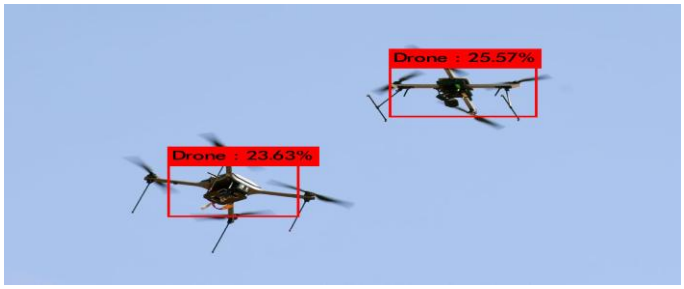

Figure 6. Multiple Drones Detected on an Image from India Aviation-Show at 800 Epochs of Training

The following result shows a detection at a restricted area scene.

Figure 7. Drone Detected on an Image of a Restricted Area

Generally, a deep learning network will consume several hours or days even weeks for getting some meaningful training to give high precision in detection even at the very higher end hardware such as GPU and TPU clouds. But the scope of this work is to complete the training process within the limitations of a conventional CPU.We observed that the training of YOLOv2 with two images consumed less than 30 minutes and YOLOv3 with two images consumed less than 20 minutes. Hence, we proved and demonstrated that it is possible to train a deep learning network with in an hour and get some meaningful testing / detection performance in terms of mAP. From this we know that anyone who is starting a deep learning research and trying to implement a complex deep learning system can really able to complete a simple prototype if they try to do the initial experiments with very low number of samples (images or any data).

V. CONCLUSION

In this work, we successfully implemented a drone detection system with two different Tiny YOLO models and demonstrated the possibility of training a deep learning network with insignificantly small number of training images. In our experiment, we have only used 2 drone images to train the Tiny YOLOv1 and Tiny YOLOv2 network and achieved an acceptable detection performance. The performance of detection at different levels of training (training epochs) is analyzed and we found that even with only two images a deep learning network gets trained in a progressive way. The evaluation made on each 100 of the epochs of training shows a significant improvement in detection performance with respect to the increase in number of epochs of training. Since there are only two images for training, the training process end up with 'over fitting' problem after around 800 epochs of training. It means that, a maximum of 800 or even 700 epochs of training are sufficient if we use only two images for training.

Most amazingly, the trained network was capable of detecting drones even from completely different images. To prove that, we chose some of the random images from[15] and from Internet and showed successful detection of drones in them.

Our results proved that any research on deep learning can be done on a normal computer without any sophisticated hardware such as GPU and TPU and giving solutions for the hardware related obstacles in front of Deep Learning Research. Generally, even with GPU and TPU clouds, the training process of deep learning will consume several days or even months since there will be a huge dataset for training. In our work, we showed the possibility of training a huge deep learning network on an insignificant hardware (CPU) with a dataset of insignificant size

REFERENCES

1. Jason Brownlee, "A Gentle Introduction to the Challenge of raining Deep Learning Neural Network Models, February 15, 2019 in Deep Learning Performance, machinelearningmastery.com
2. Jason Brownlee, "Impact of Dataset Size on Deep Learning Model Skill and Performance Estimates", January 2, 2019 in Deep Learning Performance, machinelearningmastery.com
3. Michael Chui et al., "Notes from the AI frontier: Insights from hundreds of use cases, McKinsey Global Institute, Discussion Paper, April 2018
4. Arne Schumann, Lars Sommer, Johannes Klatte, Tobias Schuchert, Jurgen Beyerer, "Deep Cross-Domain Flying Object Classification for Robust UAV Detection," IEEE August 2017.
5. Gao Q, Parslow A,Tan M , "Object Motion Detection Based on Perceptual Edge Tracking", IEEE 2016.
6. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "Unified, Real-Time Object Detection", IEEE 2016.
7. Cemal Aker, Sinan Kalkan, "Using Deep Networks for Drone Detection", IEEE July 2017.
8. Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas and Vladimir Golkov, "FlowNet: Learning Optical Flow with Convolutional Networks", IEEE 2015.
9. Xinghua Li, Qinglei Chen and Haiyang Chen, "Detection and Tracking of Moving Object Based on PTZ Camera", IEEE 2012.
10. Zhihui Li, Haibo Liu and Di Sun, "Moving Object Detection and Locating Based on Region Shrinking Algorithm", IEEE 2012.
11. Rajkumari Bidyalakshmi Devi and Khumanthem Manglem, "A Survey on Different Background Subtraction Method for Moving Object Detection", IEEE 2016.
12. Muhammad Saqib, Abin Sharma, Sultan Daud Khan and Michael Blumenstein, "A Study on Detecting Drones Using Deep Convolutional Neural Networks", IEEE 2017.
13. Mingyang Yang, A "Moving Objects Detection Algorithm in Video Sequence", IEEE 2014
14. Christian Reiser, "Bounding box detection of drones (small scale quadcopters) with CNTK Fast R-CNN", https://github.com/creiser/drone-detection
15. Chuan-en Lin, "drone-net", https://github.com/chuanenlin/drone-net
16. Alexey, "Windows and Linux version of Darknet Yolo v3 & v2 Neural Networks for object detection", https://github.com/AlexeyAB/darknet
17. Joseph Redmon, "Darknet: Open Source Neural Networks in C", http://pjreddie.com/darknet/,2013-2016
18. Miasnikov, E., "Threat of Terrorism Using Unmanned Aerial Vehicles: Technical Aspects", Center for Arms Control, Energy and Environmental Studies, Moscow Institute of Physics and Technology, Moscow, 2015.
19. Humpreys, T., "Statement on the Security Threat Posed by Unmanned Aerial Systems and Possible Countermeasures", Statement to the Subcommittee on Oversight and Management Efficiency of the House Committee on Homeland Security, 18 March 2015, Washington D.C., 2015.

20. Dinesh Sathyamoorthy, "A Review of Security Threats of Unmanned Aerial Vehicles and Mitigation Steps", Article Published at ResearchGate.net, October 2015.
21. Rangpum Hamatapa, Dr.Charoen Vongchumyen, “Image Processing for Drones Detection”, 978-1-7281-0067-8/19, IEEE,2019.
22. Suresh Arunachalam. T, Shahana. R, Vijayasri. R, Kavitha. T, “Flying Object Detection and Classification using Deep Neural Networks”, International Journal of Engineering Trends and Technology (IJETT) - Volume 67 Issue 3- March 2019
23. Deep Learning on Small Datasets without Pre-Training using Cosine Loss, Bjorn Barz ,Joachim Denzler, arXiv:1901.09054v1 [cs.LG] , 2019
24. Evaluation of the Performance of Tiny YOLOv3 based Drone Detection System with Different Drone Datasets, Manuscript sent for Journal Publication.

AUTHORS PROFILE

Ms.T.Kavitha is a Research Scholar in the Department of Computer Science and Engineering, Periyar Maniammai Institute of Science & Technology (PMIST) (Deemed to be University), Vallam, Thanjavur, Tamil Nadu, India. She completed her B.E. from Bharathidasan University and M.Tech from Anna University. Presently working as an Assistant Professor (SS) in the Department of CSE, PMIST. She has taught many subjects in CS. She is a life member in ISTE and CSI. She published 12 papers in conferences and journals. Her research interests include image and video processing, machine learning, and deep neural networks.

Dr.K.Lakshmi is working as a professor in Department of CSE, Periyar Maniammai Institute of Science & Technology (Deemed to be University), Vallam, Thanjavur, Tamil Nadu, India. She received her Ph.D. from Anna University, Chennai. She is a life member in ISTE and CSI. She has 20+ years of experience including teaching and IT industry experience. She has more than 25 journal publications and 30+ conference publications. Her areas of interest include Machine Learning, Text Mining, Internet of Things, Information (Image/Text) Retrieval.