JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

OBJECT DETECTION AND RECOGNITION IN VIDEOS USING DEEP LEARNING

Prof. Shwethashree GC[1], V Harshith[2], Vachan Patil[3] , Vikash Kumar[4] , Mohith K S[5]

[1]Professor, Department of Computer Science and Engineering, SJCE, Mysuru, Karnataka, India
[2]Student, Department of Computer Science and Engineering, SJCE, Mysuru, Karnataka, India
[3]Student, Department of Computer Science and Engineering, SJCE, Mysuru, Karnataka, India
[4]Student, Department of Computer Science and Engineering, SJCE, Mysuru, Karnataka, India
[5]Student, Department of Computer Science and Engineering, SJCE, Mysuru, Karnataka, India

Abstract: Object detection and recognition are the most basic and central tasks in computer vision. Its task is to find all the interesting objects in the image, and determine the category and location of the objects. Object detection and recognition are widely used and have strong practical value and research prospects. With the development of convolutional neural networks in recent years, significant breakthroughs have been made in object detection. Still Deeper neural networks are more challenging to train. We present a residual learning framework to ease the training of networks that are substantially deeper when compared to previously used ones.

IndexTerms - Artificial intelligence, machine learning, deep learning, convolutional neural network, object detection, image recognition, video processing, and Mobile Net.

1. INTRODUCTION

With the rapid development of modern technology and the popularization of social media and self-media, a large amount of visual information has followed. The images and videos have become an important carrier of a vast network of information. They bring us a convenient way of recording and sharing information about different objects, but it's difficult for us to classify and recognize the gathered information directly. Hence, to end this complication, the use of the computer to classify and recognize data or objects in these images and videos has gained significant prominence.

2. LITERATURE SURVEY

2.1 Image Recognition Technology Based on Machine Learning -Lijuan Liu1, Yanping Wang, and Wanle Chi

In this paper, the application of image recognition technology based on machine learning in license plate recognition is studied. Image recognition refers to the technology of using computers to process, analyze and understand the image to identify different patterns of targets and objects. It is a practical application of a deep learning algorithm. Here some basic technologies of license plate recognition are studied, such as image processing, pattern classification, machine learning, artificial intelligence, and so on.

2.2 You Only Look Once: Unified, Real-Time Object Detection - Yingxu Wang, Tony Cai, and Omar Zatarain

YOLO is a new approach to object detection. Previous work on object detection repurposes classifiers to perform detection. But here they frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. Here a single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network here, it can be optimized end-to-end directly on detection performance.

2.3. Objects Detection and Recognition in Videos for Sequence Learning -Yingxu Wang, Tony Cai, and Omar Zatarain

A key challenge to sequence learning for video comprehension is object detection and localization in a dynamic and real-time environment. This paper presents two methodological approaches to autonomous and generic object detection and localization in video sequences. A set of experiments and case studies has been conducted for practical video image processing and is demonstrated for sequence learning. This work paves a way to sequence learning toward enhanced computer and robot vision technologies.

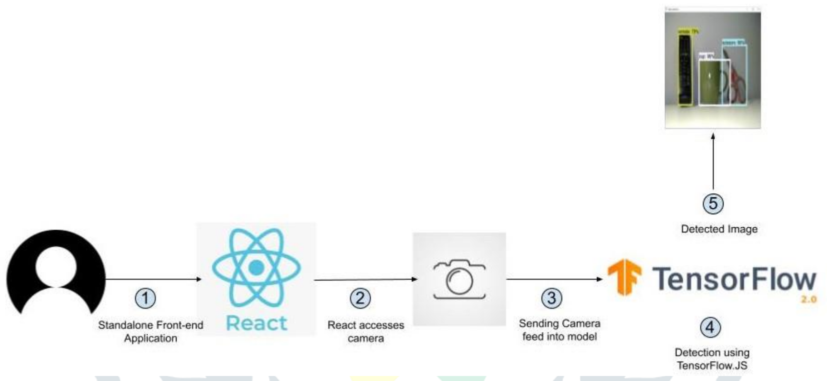
3. METHODOLOGY

Object detection refers to the capability of the device to locate objects in images or videos and identify them. We plan to train a neural network on the COCO dataset which has similar layers to MobileNet. A convolution neural network algorithm is used to improve the ability to classify and recognize two-dimensional images and videos, speed up the convergence of the algorithm, reduce the number of iterations and shorten the training period, and achieve good classification results.

The algorithm uses different stages to improve the performance of the network, reducing loss function in the network and improving the accuracy of the feature extraction of the network as mentioned below. Firstly, a recurrent neural network is introduced into the convolutional neural network and the deep features of the image are learned in parallel using the convolutional neural network and the recurrent neural network. Secondly, according to the MobileNet neural network, a similar architecture is constructed. Then the COCO dataset is passed through the model and verified with the same dataset. This model sends and receives the information from a web application that acts as a platform.

JETIR

4. SYSTEM DESIGN



1
Standalone Front-end Application
React
2
React accesses camera
3
Sending Camera feed into model
TensorFlow
2.0
4
Detection using TensorFlow.JS
5
Detected Image

Figure 1: Data Flow Diagram

4.1 Import Modules

This process begins by initially importing all the necessary modules like OpenCV, TensorFlow, Node JS, React, etc.

4.2 Creating a standalone Front-end application

After importing all the necessary modules, we create a standalone Front-end web application using React for accessing camera footage, displaying predicted information, and for easier usage in any type of device.

4.3 Sending a camera feed into the model

The live video footage is fed from the camera into the model. The model then tries to detect multiple objects in the frame based on its confidence value.

4.4 Make Detection using TensorFlow.JS Model

A machine learning model is trained and converted using Tensorflow.js. Tensorflow.js model directly detects the camera feed and detects different objects based on the adjusted weights of the model. Tensorflow.js is an ML library for JavaScript. It helps in the deployment of machine learning models directly into node.js or a web browser.

4.5 Displaying the detected objects

After detecting the objects, the classified object is differentiated by the colored square and the name of the object on the left corner of the square.

5. SYSTEM ARCHITECTURE

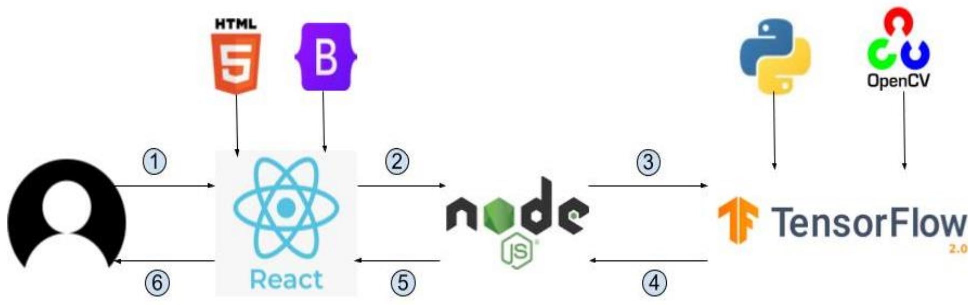


Figure 2: Architecture Diagram

5.1 User Interaction with Frontend

React is a JavaScript library that will be used for building user-friendly interfaces. It is designed to solve the issues of partial content updates for web pages, which can usually be found in the development of one-page applications. Compared to other frontend frameworks, the React code is easier to maintain and is flexible due to its modular structure. This flexibility will help us to save a huge amount of time.

5.2 Requesting Connection from Frontend to Backends

Next, we will connect the React frontend to a backend. For that, we will use Node JS which is primarily used for non-blocking, event-driven servers, due to its single-threaded nature. It's used for traditional websites and back-end API services but was designed with real-time, push-based architectures in mind. To make this connection we just have to fetch data from the port the backend server is located on.

5.3 Connecting Backend with TensorFlow.js Model

Here we will deploy a Tensorflow.js model using NodeJS. Tensorflow.js is an ML library for JavaScript. It helps to deploy machine learning models directly into node.js or a web browser.

5.4 Fetching Machine Learning Model TensorFlow.js

Now we will use TensorFlow to train a custom object-detection model in Python, then put it into production, and run real-time inferences in the browser through TensorFlow.js.

5.5 Processing the user request by TensorFlow.js Model

TensorFlow.js is used for saving and loading models that have been created with the Layers API or converted from existing TensorFlow models. When we run it on Node.js we also have direct access to the filesystem and can save models there.

5.6 Fetching the required output

The video feed is processed by the Tensorflow.js model which detects different objects and creates colored boxes for the different objects along with the label.

6. SYSTEM IMPLEMENTATION

6.1 Data Preprocessing

Firstly the model is trained on a known image database. For this, we have used the coco dataset which contains images in 80 different categories with more than 330k images including 200k labeled images. For our use, we pick a very popular deep learning model which is used widely and will alter its layers as per our requirement using transfer learning.

6.2 Implementation of Transfer Learning on MobileNet

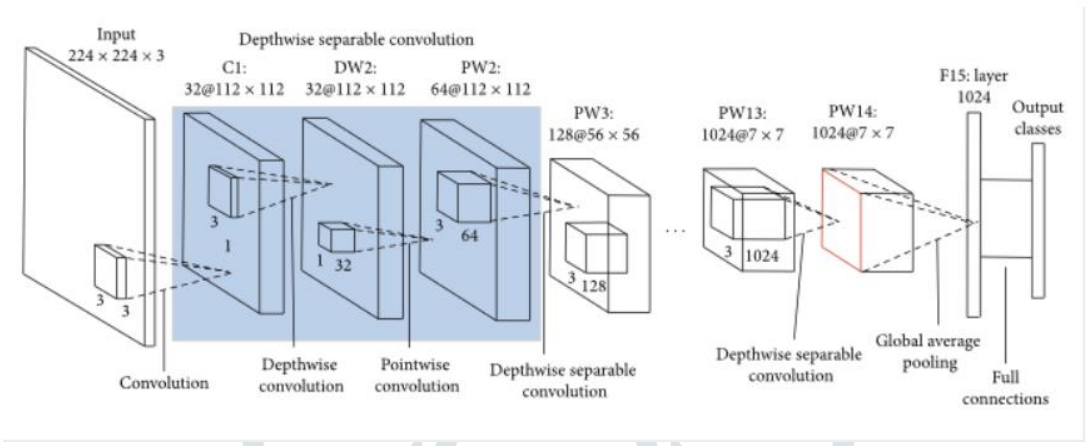
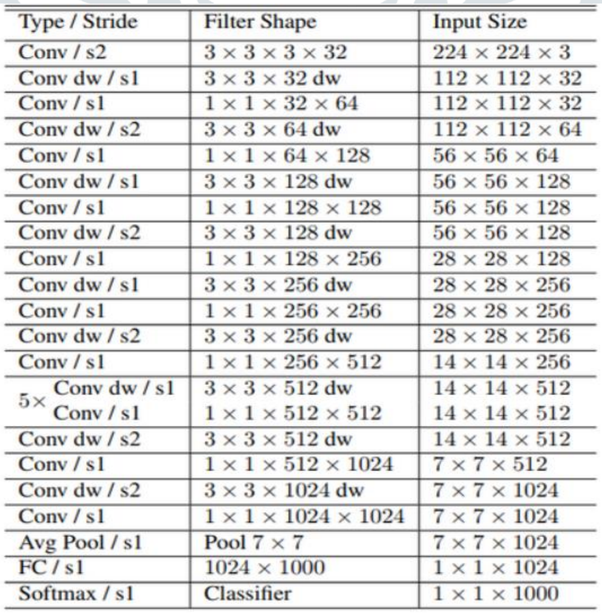


Figure 3: MobileNet50 structure

The MobileNet model is designed to be used in mobile applications. It is also TensorFlow's first mobile computer vision model. MobileNet uses depthwise separable convolutions. This significantly reduces the number of parameters when compared to the network with regular convolutions with the same depth in the nets. Which results in lightweight deep neural networks. A depthwise separable convolution is made from two operations.

- Depthwise convolution.
- Pointwise convolution.

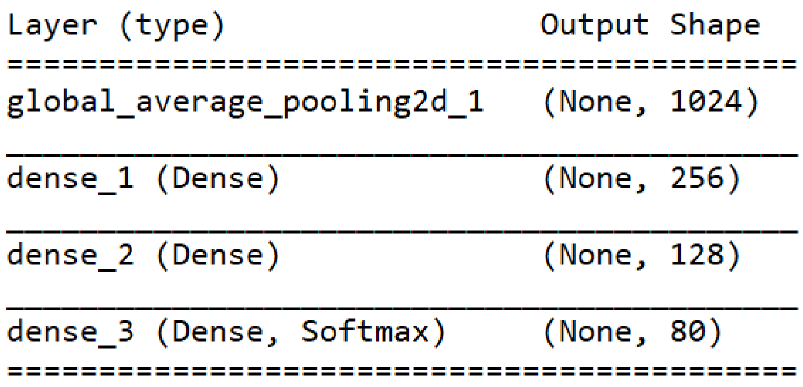
MobileNet is a class of CNN that was open-sourced by Google, and therefore, this gives us an excellent starting point for training our classifiers that are insanely small and insanely fast.



Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 4: Architecture of MobileNet

For the implementation, the last layer in MobileNet is removed and three additional dense layers are added. This ensures that the model is accurate as per our dataset and the last layer has a classifier that classifies the input image into 80 categories as per the dataset we have used to train the model.



```
Layer (type)                 Output Shape
=========================================
global_average_pooling2d_1   (None, 1024)
_________________________________________
dense_1 (Dense)              (None, 256)
_________________________________________
dense_2 (Dense)              (None, 128)
_________________________________________
dense_3 (Dense, Softmax)     (None, 80)
=========================================
```

Figure 5: Extra layers added to MobileNet after pooling layer

6.3 Web Interface Implementation

An interface was developed using Javascript, CSS, and HTML. This combination was hosted on a React Application which deploys our interface on a port and starts listening and monitoring for any requests.

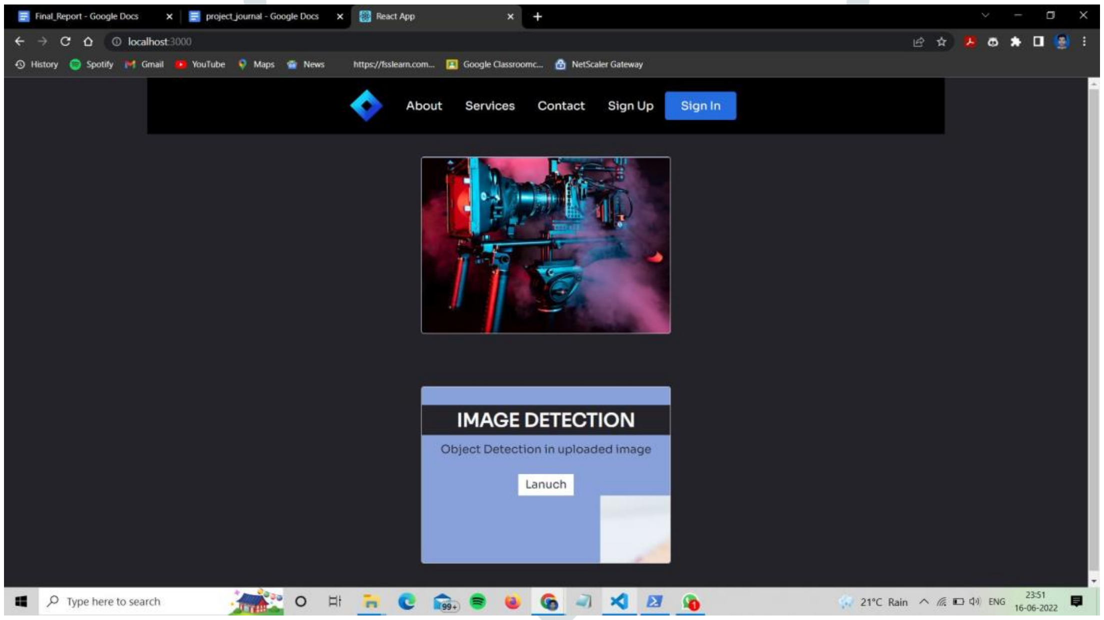


Figure 6: Interface Screen

\

7. SYSTEM TESTING AND RESULT ANALYSIS

We developed and trained all the architectures based on the steps mentioned in the System implementation section. We plotted graphs at the final step of each architecture that we implemented. Various test cases were developed to check the functionality of our web page and all the test cases were provided with appropriate output.

7.1 Transfer Learning model on MobileNet

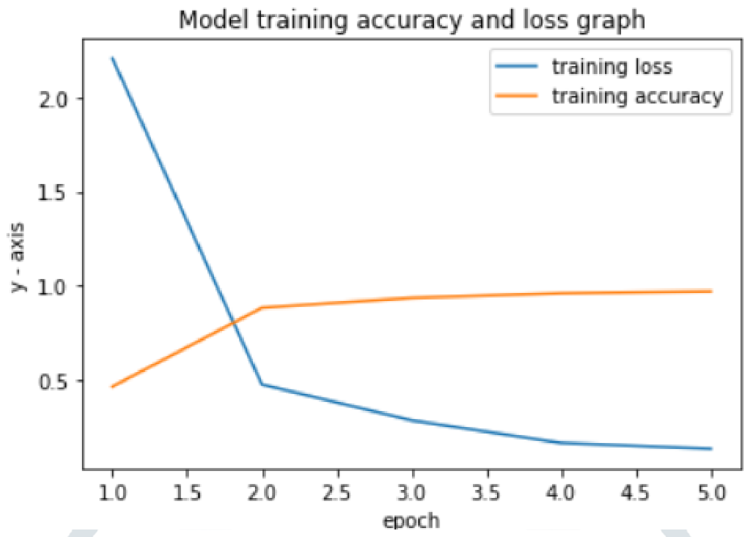


Figure 7: Interface Screen

While training the accuracy of the model reaches 0.96 in 5 epochs.

On testing the model overall with a random dataset it was able to achieve an accuracy of 0.91 which is slightly less than the training accuracy.

7.2 Interface Testing

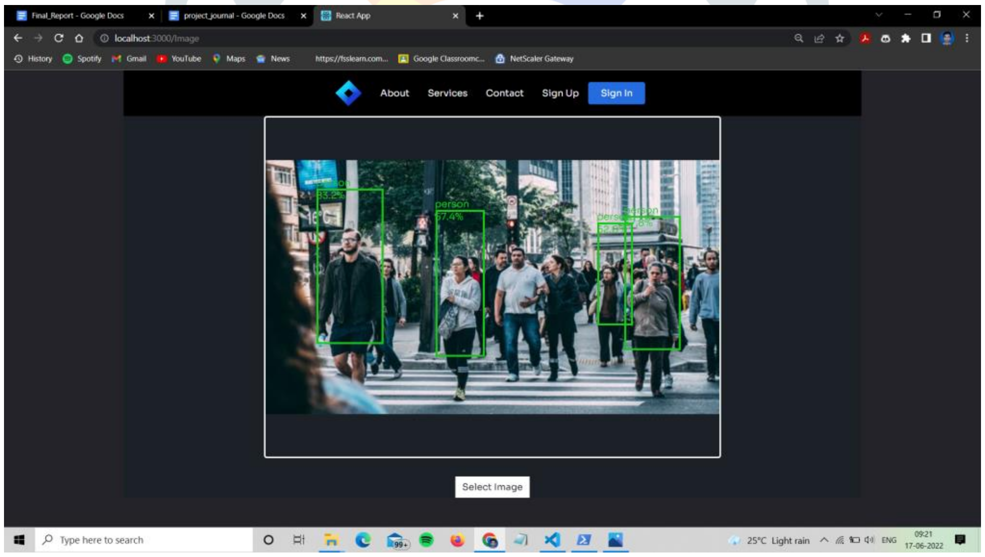


Figure 8: Interface Test on the Image

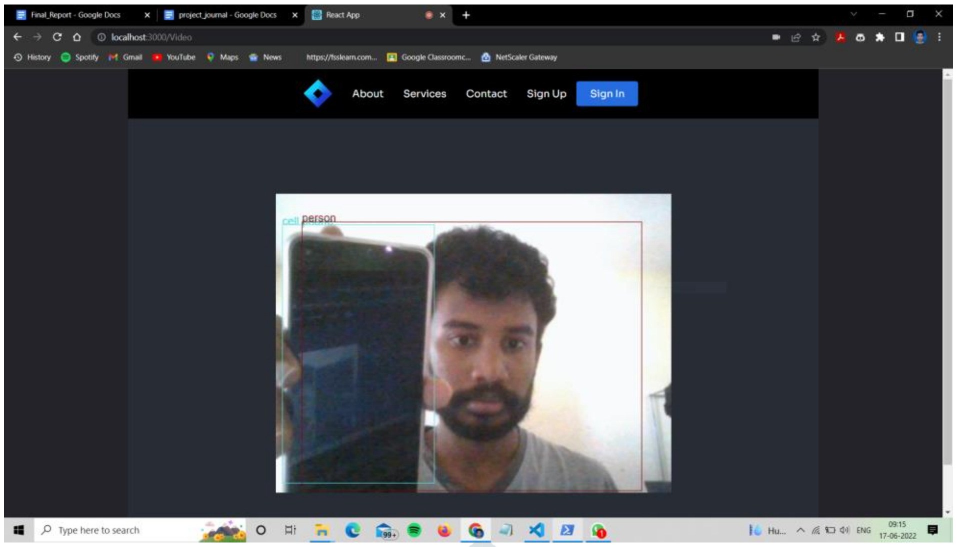


Figure 9: Interface Test on a Real Video

8. CONCLUSION AND FUTURE WORK

In this work, we improve the ability of the convolutional neural network to classify and recognize two-dimensional images and speed up the convergence of the algorithm.
To summarize the project in a few short points,
i. MobileNet50 was studied and implemented.
ii. Observing the limitations of neural architectures was explored.

Video object detection is both an important research topic and a complex problem in practical applications. As video object detection is a popular and promising field in the field of machine learning, multiple object detection algorithms have been established, and the demands for video data processing have gradually increased. But at present, the computational loads of the existing algorithms and the ability to achieve real-time performance levels are the biggest constraint as they restrict the applications of these algorithms.

Also detecting object motion in a real-world situation is one of the challenges we face since object detection here is a continuous process. During the detection defocusing is caused because the imaging device fails to always focus on the target accurately. In defocus conditions, target objects will appear unclear and blurry. Therefore, the features extracted by the network are also less clear, making such objects difficult to detect. As a future work potential improvements could be included when detection algorithms become mature, thereby increasing the accuracy and performance of video object detection.

REFERENCES

[1] Lijuan Liu, Yanping Wang, and Wanle Chi, “Image Recognition Technology Based on Machine Learning”.
[2] Xiao H, Rasul K, and Vollgraf R. Fashion-MNIST: “A Novel Image Dataset for Benchmarking Machine Learning Algorithms”.
[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep Residual Learning for Image Recognition”.
[4] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun., “Object detection networks on convolutional feature maps”.
[5] Licheng Jiao, Ruohan Zhang, and Fang Liu, “New Generation Deep Learning For Video Object Detection”.
[6] L. Jiao et al., “A survey of deep learning-based object detection”.
[7] Joseph Redmon, Santosh Divvala, and Ross Girshick, “You Only Look Once: Unified, Real-Time Object Detection”.
[8] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective”.
[9] Bender, E.A. (1996), “Mathematical Methods in Artificial Intelligence”.
[10] Viola, P. and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features, IEEE Proceedings of Computer Vision and Pattern Recognition”.
[11] S. Gidaris and N. Komodakis, “Object detection via a multi-region & semantic segmentation-aware CNN model”.
[12] R. Girshick, “Fast R-CNN”.
[13] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”.
[14] Wang, Y., O. Zatarain, T. Tsai, and Daniel Graves, “Sequence Learning for Image Recognition in Videos with Differential Neural Networks”
[15] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. “Decaf: A deep convolutional activation feature for generic visual recognition”