iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 7 **Issue:** IV **Month of publication:** April 2019

DOI: https://doi.org/10.22214/ijraset.2019.4580

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 7 Issue IV, Apr 2019- Available at www.ijraset.com

Detection of Distracted Driver

S. Ghouhar Taj[1], Sk. Arshiya yasmine[2], C. Chaitra[3], C. Subramanyam Reddy[4]
[1]*Adhoc Lecturer, Dept of CSE, JNTUACEP, Pulivendula, AP, India*
[2, 3, 4]*Student, Dept of CSE, JNTUACEP, Pulivendula, AP, India*

Abstract: ***In most of the cases, the prime cause of an accident is the distracted state of driver. With the increase of In-Vehicle Information System such cases are increasing. This problem can be solved by monitoring and predicting the state of driver while driving and installing the autonomous prevention system to take preventive actions. In order to realize this strategy we have used Convolution Neural Networks and deep learning concepts in order to classify the image of driver into 10 different classes which include texting, talking to passenger, talking on the phone, looking left or right, searching something at the back seat, hair and makeup, operating radio and drinking.***
Keywords: Convolutional Neural Network, Distraction, Driver, Deep learning, Classification, Detection.

I. INTRODUCTION

According to the CDC motor vehicle safety division, there are 3,287 deaths each day due to fatal car crashes. On average, 9 of these daily fatalities are related to distracted driving. This accounts for approximately **25%** of all motor vehicle crash fatalities. Driver distraction is reported to be responsible for more than **58%** of teen crashes.

State Farm hopes to improve these alarming statistics, and better insure their customers, by testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviours. In this project, we have created and refined machine learning models to detect what the driver is doing in a car. This is done by predicting the likelihood of what the driver is doing in each picture.

The input to our models are images of people driving. Each image belongs to one of the ten classes described later. We then use two different types of convolutional neural networks (CNN): VGG-16 and GoogleNet to predict to which class the given images belong. The output is a list of predicted class labels.

II. RELATED WORK

This problem was a public challenge hosted on Kaggle by State Farm insurance company two years ago [2]. Some of the solutions were based on SVM model that detect the use of mobile phone while driving [3]. Others were based on face and hand segmentation using RCNN [4]. Some approaches included the use of handcrafted features (HOG and BoWs) [5]. There are quite a few approaches based on Deep CNN models which are pre-trained on ImageNet such as AlexNet, ResNet-152, VGG-16. Some solutions consist of genetically-weighted ensemble of convolutional neural networks.

III. DATASET DESCRIPTION

State Farm is a large group of insurance and financial services companies throughout the United States. They released their dataset of 2D dashboard camera images for a Kaggle challenge. The dataset had 22400 training images and 79727 testing images. Resolution was 640 x 480 pixels.The training images had corresponding labels attached. Labels belonged to one of the ten classes as mentioned below:

1) c0: normal driving
2) c1: texting - right
3) c2: talking on the phone - right
4) c3: texting - left
5) c4: talking on the phone - left
6) c5: operating the radio
7) c6: drinking
8) c7: reaching behind
9) c8: hair and makeup
10) c9: talking to passenger

The training set consists of 22400 images which are split into 2 parts i.e. train and validation sets.

©IJRASET: All Rights are Reserved

Below are the sample images of the input.



IV. TECHNICAL APPROACH

Deep CNN is essentially a kind of Artificial Neural Network (ANN) which is propelled by the animal visual cortex. Since most recent couple of years, CNNs have indicated amazing advancement in different assignments like object detection, image classification, natural language processing and some more.

A. CNN Architecture

In AI, a convolutional neural system (CNN, or ConvNet) is a class of deep, feed-forward artificial neural systems that has effectively been applied to analyse visual imagery.

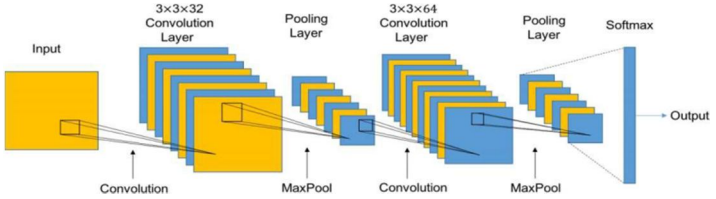


Fig.1: Basic CNN Architecture

A CNN comprises of an input and a output layer, just as numerous hidden or concealed layers. A CNN design is shaped by a pile of unmistakable layers that change the information volume into a yield volume through a differentiable function.

B. Convolutional Layer

The core building block of a CNN is the convolutional layer. The layer's parameters comprise of a lot of learnable filters, which have a little receptive field, yet reach out through t he full depth of the input volume. A two dimensional activation map of each filter is produced and it is convolved across the width and height of the input volume. Therefore, the network learns filters that activate when it identifies some particular kind of feature at some spatial position in the input.

C. Pooling Layer

Pooling is a type of non-linear down-sampling. Max pooling is the most common implementation of pooling. The input picture is partitioned into a set of non-overlapping rectangles and for each rectangle, it outputs the maximum value. The instinct is that the careful area of a feature is less essential than its unpleasant area in respect to different features. The pooling layer works independently on each depth slice of the input and resizes it spatially. It is often inserted between successive convolutional layers. The depth of the input stays unaltered.

D. Fully Connected Layer

At last, after a few convolutional and max pooling layers, the high level reasoning in the neural networks is done by means of fully connected layers. In fully connected layer, neurons are connected to all activations in the previous layer, as found in regular neural networks.

E. Classification Layer

The classification layer determines how the deviation between the predicted and true labels is penalized by training and is generally the final layer. Softmax loss is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values ranging from 0 to 1. Several loss functions that are appropriate for different tasks might be chosen accordingly.

©IJRASET: All Rights are Reserved

F. Transfer Learning

Transfer learning gives the chance to adapt a pre-trained model to new classes of data . In this technique a pre-prepared model is used as an initialization or fixed feature extractor as opposed to training a CNN from scratch . The following are two kinds of transfer learning systems.

1) *ConvNet as Fixed Feature Extractor:* In this technique, a pre-prepared network is introduced and the last fully connected layer is removed. this network is then treated as a fixed feature extractor. When these fixed features are extracted, the last layer is trained with these features. As shown below in Fig.2 instantiate the convolutional part of the VGG16 model. Execute the model on training and validation data once, recording the output in two numpy arrays. Then a small fully connected model is trained on top of the stored features.

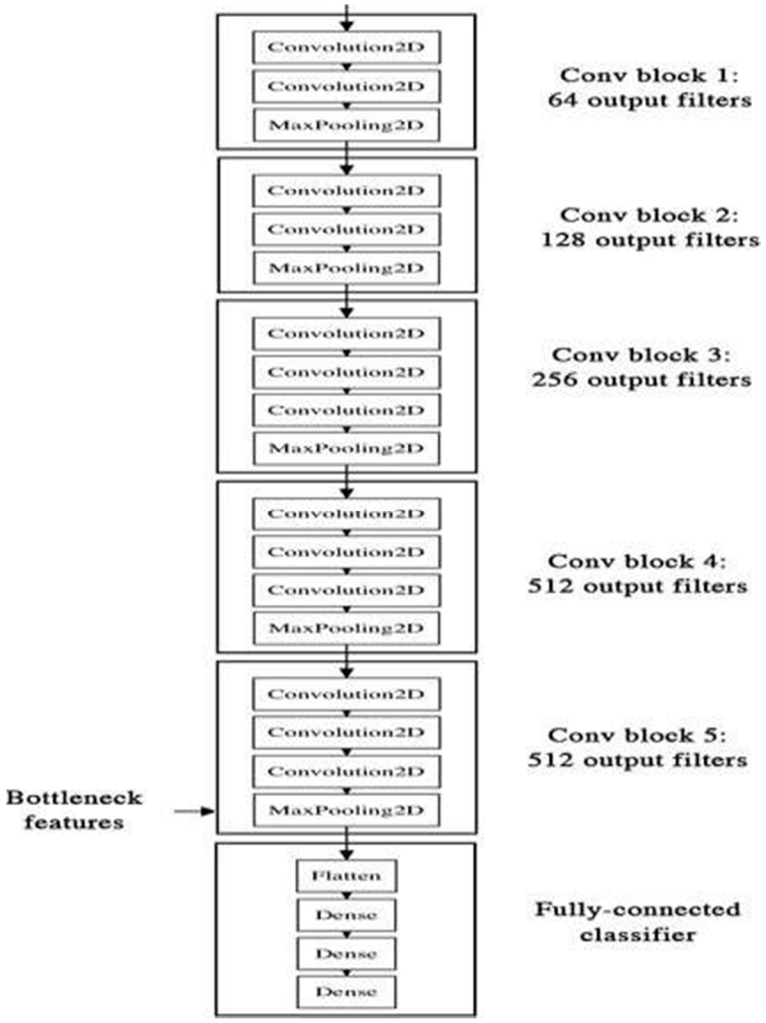


Fig.2 (File:vgg16)

2) *Fine-tuning the ConvNet:* In this technique, the weights of the pre-trained network are also fine-tuned, in addition to replacing and retraining only the classifier. It is possible to fine-tune all the layers of the ConvNet, or it's possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a ConvNet contain more generic features that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset. As shown below in Fig.3 instantiate the convolutional base of VGG16 and load its weights. Add previously defined fully-connected model on top, and load its weights. Freeze the layers of the VGG16 model up to the last convolutional block. Fine-tune the model. This is done with a very slow learning rate, and typically with the SGD optimizer rather than an adaptive learning rate optimizer such as RMSProp. This is to make sure that the magnitude of the updates stays very small, so as not to wreck the previously learned features.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 7 Issue IV, Apr 2019- Available at www.ijraset.com

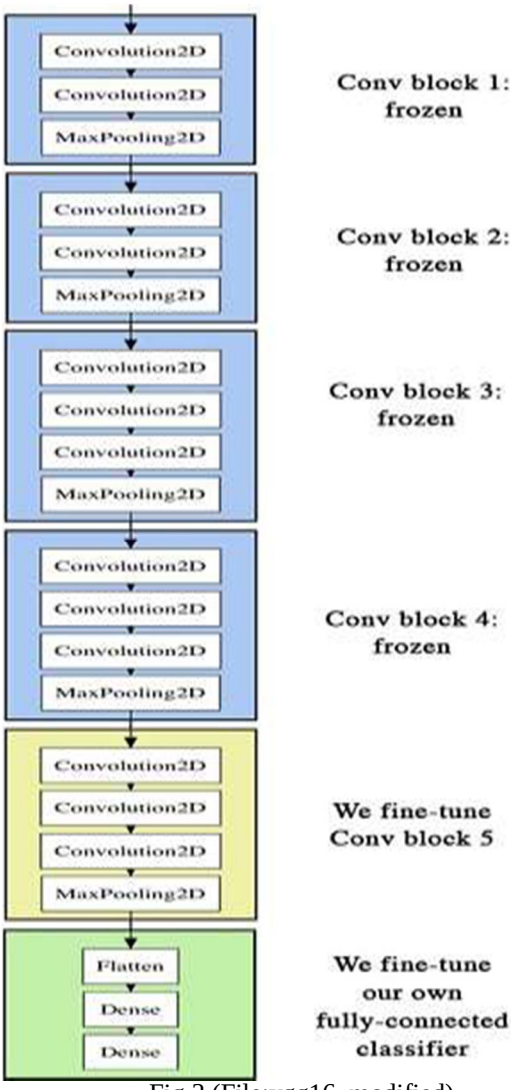


Fig.3 (File:vgg16_modified)

Coding was completed using Python and Keras with Tensorflow as the backend. Amid the training process huge memory usage was seen when every image is loaded and pre-processed. Subsequently this step was executed multiple times changing the capacity of the compute instances to account for huge RAM.

G. Refinement

To get the initial result simple CNN architecture was built and evaluated. This resulted in a decent loss. The public score for the initial simple CNN architecture(initial unoptimized model) was 2.67118.

After this to further improve the loss, transfer learning was applied to VGG16 along with investigating 2 types of architectures for fully connected layer. Model Architecture1 showed good results and was improved further by using the below techniques

1) Drop out layer was added to account for overfit ting.
2) Xavier initialization was used instead of random initialization of weights
3) Zero mean was ensured by subtracting 0.5 during Pre-processing.
4) Training was carried out with 400 epochs and with a batch size of 16

To further improve the loss metric, VGG16 along with Model Architecture1 was selected and fine-tuning was applied. SGD optimiser was used with very slow learning rate of 1e-4. A momentum of 0.9 was applied with SGD.

©IJRASET: All Rights are Reserved

H. VGG-16 Model

We setup our pre-trained VGG-16 model with the last output layer removed. We first run a forward propagation through the VGG-16 model and save the results for future use. Then, we add and train the output layer with batch size of 64, learning rate 0.001, epoch of 15, and dropout of 0.8. The dropout layer will randomly drop 80% of the neurons during the training process for each iteration, and proves significantly reduce the overfitting issue in our experiments. The model converges quickly in about 10 epoches and then stabilize throughout the training process. We plot the accuracy for both training and valid set, and the maximum accuracy in validation set is 85% with a log loss of 0.45.

The fine-tuned model was finally selected as that provided the best score and rank in the Public Leadership board. The public score for fine-tuned model(final optimized model) was 1.26397. This is best score than that of the initial simple CNN architecture(initial unoptimized model) score i.e. 2.67118.

V. RESULTS

A. Model Evaluation and Validation

During model creation and development, a validation set was used to evaluate the model. The comparison of the Public Scores for all the model architectures considered for this data set is shown in Fig.4

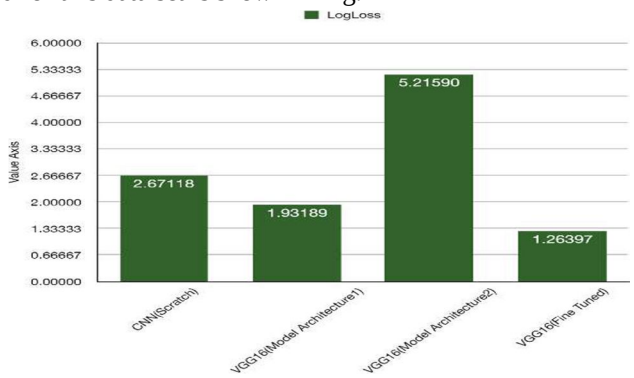


Fig.4

The final architecture chosen was the VGG16 fine-tuned model. This architecture along with hyper-parameters were chosen as they performed the best among all model combinations.

Following are the details of the final model parameters and training process:

1) VGG16 is instantiated and first 15 layers were frozen.
2) The last Conv layer i.e. Conv block 5 is fine tuned.
3) Our own layer(Global Average Pooling + Fully Connected layer) is added and fine tuned
4) Fine tuning is carried out with very slow learning rate and SGD optimizer
5) Training is carried out for 10 epochs with a batch size of 16

Finally the model is tried on the test data set. This resulted in Public Score of 1.26397. This can result in rank of 617 out of 1440 in Public Leaderboard i.e. in top 42.84%. It was observed the loss on validation dataset was 0.00751.This in caparison with public score on test dataset would suggest that there is overfitting. In order to resolve overfitting we need to consider adding/increasing the drop out and L2 regularization.

VI. CONCLUSION

The table below summarizes the overall performance for each of our models.

Table1.Model Performance Summary

Models Validation	Validation Accuracy	Validation Loss	Kaggle Score
FC	11.4%	6.1	6.8
Basic CNN	74.3%	1.2081	1.32
VGG-16	85.01%	0.4954	0.64

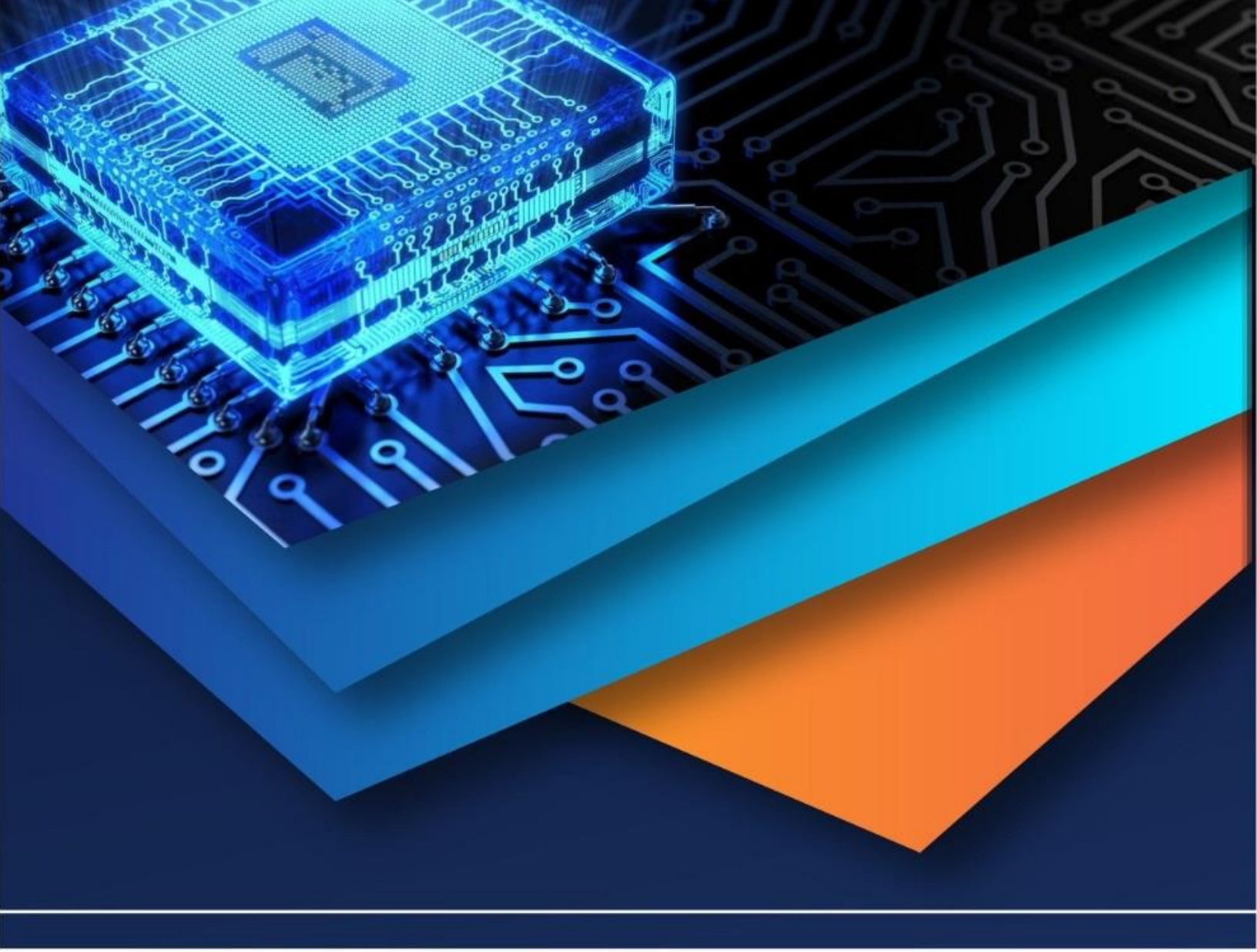
©IJRASET: All Rights are Reserved

The fine-tuned model was finally selected as that provided the best score and rank in the Public Leadership board. The public score for fine-tuned model (final optimized model) was 1.26397. This is best score than that of the initial simple CNN architecture (initial unoptimized model) score i.e. 2.67118.

REFERENCES

[1] Report on Road Accidents in India 2016-Ministry of Road Transport & Highways (MoRTH), Government of India pp. 1-2 http://morth.nic.in/showfile.asp?lid=2904
[2] Kaggle. A brief summary https://www.kaggle.com/c/state-farm-distracted-driver-detection
[3] Yehya Abouelnaga, Hesham M. Eraqi, and Mohamed N. Moustafa, "Real-time Distracted Driver Posture Classification", arXiv preprintarXiv:1706.09498
[4] T. H. N. Le, Y. Zheng, C. Zhu, K. Luu and M. Savvides, "Multiple Scale Faster-RCNN Approach to Driver¨s Cell-Phone Usage and Hands on Steering Wheel Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Las Vegas, NV, 2016, pp. 46-53
[5] Hssayeni, Murtadha D; Saxena, Sagar; Ptucha, Raymond; Savakis, Andreas, "Distracted Driver Detection: Deep Learning vs Handcrafted Features", Society for Imaging Science and Technology, Imaging and Multimedia Analytics in a Web and Mobile World 2017, pp. 20-26(7)
[6] dImageNet: VGGNet, ResNet, Inception, and Xception with Keras
[7] CS231n Convolutional Neural Networks for Visual Recognition http://cs231n.github.io/convolutional-networks/
[8] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv preprint arXiv:1409.1556
[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, "Going Deeper with Convolutions", arXiv preprint arXiv:1409.4842

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)