iJRASET

International Journal For Research in
Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VI **Month of publication:** June 2022

DOI: https://doi.org/10.22214/ijraset.2022.44611

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VI June 2022- Available at www.ijraset.com

Automatic Logo Detection with Deep Region-based Convolutional Networks

G Naga Sujini
Department of CSE, MGIT

Abstract: Deep learning is an important element of data science, which includes statistics and predictive modeling. It is extremely beneficial to data scientists who are tasked with collecting, analyzing and interpreting large amounts of data. It makes this process faster and easier. We present a deep learning system for automatic logo detection in real world images. We base our detector on the popular framework of Faster R-CNNand compare its performance to other models such as Mask R-CNN or RetinaNet. We perform a detailed empirical analysis of various design and architecture choices and show how these can have much higher influence than algorithmic tweaks or popular techniques such as data augmentation.

We also provide a systematic detection performance comparison of various models on multiple popular datasets including FlickrLogos-32, TopLogo-10 and recently introduced QMUL-OpenLogo benchmark, which allows for a direct comparison between recently proposed extensions. By careful optimization of the training procedure we were able to achieve significant improvements of the state of the art on all mentioned datasets. We apply our observations to build a detector to detect logos of the Red Bull brand in online media and images. Automatic logo detection and recognition is a challenging computer vision task with many applications. One of the typical purposes of such systems is measurement of brand visibility and prominence in online media or television broadcasts.

Keywords: Faster R-CNN, FlickrLogos-32, Logo detection, Mask R-CNN

I. INTRODUCTION

A logo is a graphical mark used to identify a company, organization, product or brand. Logos areused to represent a company's name, a particular product or service and are used heavily in the marketing of products and services. Logos have become an integral part of a company's identity anda well-recognized logo can increase a company's goodwill. A logo usually has a recognizable and distinctive graphic design, stylized name or unique symbol for identifying an organization. It is affixed,included, or printed on all advertising, buildings, communications, literature, products, stationery, vehicles, etc. Logo can be seen anywhere in the surrounding in our daily life, such as in the streets, supermarkets, on the products or services, on administrative documents, etc. . The last decades have seen an explosion of the amount of digitized document libraries. In order to properly index these documents, it is necessary to categorize as well asto retrieve them [1].

Automated logo detection from unconstrained "in-the-wild" images benefits a wide range of applications, e.g. brand trend prediction for commercial research and vehicle logo recognition for intelligent transportation this is inherently a challenging task due to the presence of many logos in diverse context with uncontrolled illumination, low-resolution, and background clutter. Existing logo detection methods typically consider a small number of logo classes with the need for large sized training data annotated at the logo object instance level, i.e. object bounding boxes.

Whilst this controlled setting allows for a straightforward adoption of the state-of-the-art object detection models it is unsalable to real-world logo detection applications when a muchlarger number of logo classes are of interest but limited by the extremely high cost for constructing large scale logo dataset with exhaustive logo instance bounding box labeling therefore unavailability and [2] lacking incremental model learning to progressively update and expand the model to increasingly more training data without fine-grained labeling. Existing models are mostly one-pass trained and statically generalized to new test data. In this work, we consider scalable logo detection learning in a very large collection of unconstrained images without exhaustive fine-grained object instance level labeling for model training.

II. LITERATURE SURVEY

"Deep Learning for Logo Detection" Automatic logo detection and recognition is a challenging computer vision task with many applications. One of the typical purposes of such systems is measurement of brand visibility and prominence in online media or television broadcasts. The task can be formulated as a special case of generic object detection, albeit with some significant differences.

Most notably, logo instances usually occupy only a small portion of the image, often being captured by accident, for example as sponsor logo on an athlete's sportswear or a driver's vehicle etc. There is also a different kind of visual variability, for example when the logo's color design changes depending on target background or when pictorially designed logo appears as a three dimensional sculpture. [1].

"Logo Detection Based on Convolutional Neural Networks" proposes a novel logo detection method based on the global pyramid text attention module. The global pyramid text attention module (GPTAM) contains the Global Text Attention Module (GTAM) and the Pyramid Text Attention Module (PTAM). GTAM can take use of the context information of texts in logos to reduce detection error, and PTAM can handle logos with multiple scales. We use non-maximum suppression(NMS) method in the post processing to obtain the final detection. The experiments on FlickrLogos-32 and TopLogos-10 databases demonstrate that our proposed outperforms the state-of-the-art methods. Furthermore, our method also achieves better or comparable performance on ICDAR2013 and ICDAR2015 datasets for scene text detection. [2].

"Scalable Logo Detection and Recognition with Minimal Labeling" describe a new approach to detecting and locating brand logos in an image using machine learning methods and synthetic training data. Deep learning methods, particularly the use of Convolutional Neural Networks (CNN), have been very popular for extracting visual information, such as image shapes and objects, from images. A CNN has parameters and configuration information that are learned from training images. To obtain good accuracy usually a large amount of labeled (groundtruthed) images are required for training. While existing methods rely on manually labeled images, our method is fully trained with images obtained in an automated manner with minimal human supervision. [3].

"Graphic Logo Detection with Deep Region-based Convolutional Networks" this paper, first manually collected and annotated 6,400 images and mixes them with FlickrLogo-32 dataset, forming a larger dataset. Secondly, we constructed Faster R-CNN frameworks with several widely used classification models for logo detection. Furthermore, the transfer learning method was introduced in the training process. Finally, clustering was used to guarantee suitable hyper parameters and more precise anchors of RPN.Experimental results show that the proposed framework outperforms the state of-the-art methodswith a noticeable margin. [4].

"Automatic Graphic Logo Detection via Fast Region-based Convolutional Networks" which usesthe approach that is is based on Fast Region-based Convolutional Networks (FRCN) proposed by RossGirshick, which have shown state-of-the-art performance in several generic object recognition tasks (PASCAL Visual Object Classes challenges). In particular, we use two CNN models pre trained with the ILSVRC ImageNet dataset and we look at the selective search of windows 'proposals' in the pre-processing stage and data augmentation to enhance the logo recognition rate [5].

III. DESIGN METHODOLOGYSYSTEM ARCHITECTURE:

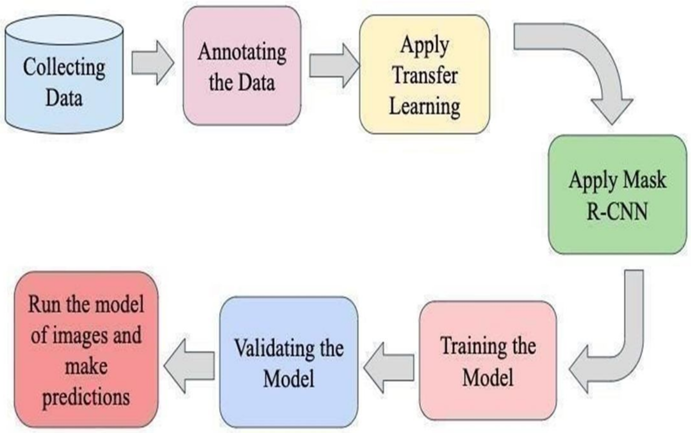


Fig 3.1 System Architecture

CNN is a segmentation model which segments at each instance and finds the location of pixels of the class .It segments different objects which are presenting an image irrespective of type of objects they are and training them on the Flickr Logo-32 dataset which we are using for the project. There are step wise procedure are to be followed to attain betterresults of CNN.

A. Data Set Collection

The images are from FlickrLogo-32 dataset, we train and validate images which are present in the dataset and implement the working of the model on the images which are taken from the dataset.

FlickrLogo-32 Datasets and Augmentation FlickrLogo-32 dataset contains 8240 natural scene images, including 32 classes, 70 images for eachclass and 6000 non logo images. To augment thedata scale for deep learning, in [12], Hang Su et al. expanded FlickrLogo-32 by randomly addingpre-processed logo templates on these non-logosubsets of the dataset. However, experiments showed that this work can hardly make anypromotion, since new added objects may be notexactly matching with surrounding contexts. Tokeep the same data distribution with FlickrLogo-32, we firstly analyze the original data to confirmcollection and annotation rules. To avoid imagerepetition, we check duplication among newcollections. Ultimately, 200 new images for eachclass are mixed with the original FlickrLogo-32,thus forming a new dataset, called Logo32-270,which contains 270 samples for each class, 8640 images in all.

B. Apply Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task [3]. Transfer Learning differs from traditional Machine Learning in that it is the use of pre-trained models that have been used for another task to jump start the development process on a new task or problem as shown in Figure 3.2.2. Transfer learning is an optimization that allows rapid progress or improvedperformance when modeling the second task.

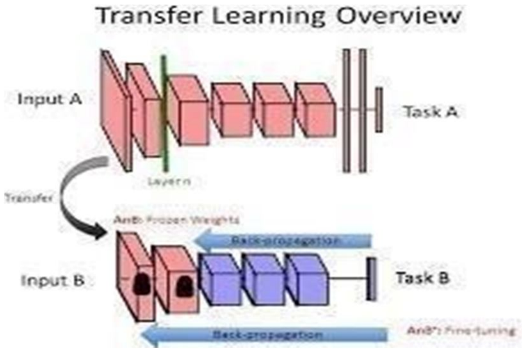


Fig 3.2 Transfer Learning

C. Apply Mask R-CNN

Mask R-CNN is a combination of a Faster R-CNN that does object detection (class bounding box) and FCN (Fully Convolutional Network) that does pixelwise boundary. Mask R-CNN is conceptually simple - Faster R-CNN has two outputs for each candidate object, a class label anda bounding-box offset; to this we add a third branchthat outputs the object mask, which is a binary mask that indicates the pixels where the object is in the bounding box. Mark R-CNN is image segmentation technique which provides easy work framework for segmentation of different objects in an image. It will generate the masks for the objectsin an image. It is used for object segmentation and generating mask for each object in each instance. Mask R-CNN is simple model which adds a third branch that outputs an object mask which makes the Mask R-CNN a simple and flexible idea.

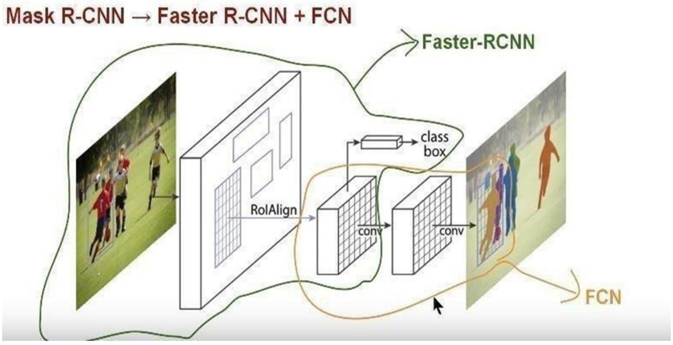


Fig 3.3 Mask R-CNN

D. Run the Model of images and makePredictions

This includes applying model to our Dataset andmaking precisions.
Precision is calculated by:
TP/ (TP+FN)
Where TP =True Positive, FN=False Negative.

1) Technique

- *Convolution Neural Networks*

CNN is image segmentation technique whichprovides easy work framework for segmentation of different objects in an image. It will generate the masks for the objects in an image. It is used for object segmentation and generating mask for each object in each instance. The importance of qualityof image has been improved largely over the years. The techniques of image segmentation CNN, R-CNN are collectively used and trained to get better roof the detection of an object in an image at an instance. These techniques are useful to get better view of objects in an image by image segmentation and instance segmentation. There are many challenging tasks in the instance segmentation as it has to detect the correct objects every time in an image when the techniques are applied and have to classify the different classes of images and train accordingly. The objective is to specify the similar objects in different images into classes and to localize each object by using the bounding segmentation. We need to order each pixel into a set of classes without differentiating the objects in different instances of the different images. Mask CNN works. In similar way compared to Faster CNN it has an extended layer ofa branch which is helpful for predicting segmentation masks on the region which we are interested in (ROI) along with a branch forclassification and for bounding box regression.

- *VGG-16 Model*

VGG-16 is convolutional neural network architecture and its name VGG-16 comes from the fact that it has 16 layers. Its layers consist of Convolutional layers, Max Pooling layers, Activation layers, fully connected layers. VGG-16 network is trained on ImageNet dataset which has over 14 million images and 1000 classes, and achieves 92.7% top-5 accuracy. It surpasses AlexNet network by replacing large filters of size 11 and 5 in the first and second convolution layers with small size 3x3 filters.

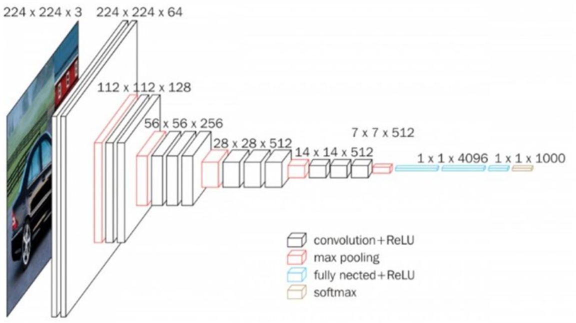


Fig 3.7 VGG-16 Architecture

As shown in the Fig 3.7, the input to conv1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very smallreceptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). Inone of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv.layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

Three Fully-Connected (FC) layers follow a stackof convolutional layers .The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalization (LRN), such normalizationdoes not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

There are 13 convolutional layers, 5 Max Pooling layers and 3 Dense layers which sums up to 21 layers but only 16 weight layers.

Conv 1 has number of filters as 64 while Conv 2 has 128 filters,Conv 3 has 256 filters while Conv 4 and Conv 5has 512 filters. This model is finished by two fully connected hidden layers and one output layer. The two fully connected layers have the same neuron numbers which are 4096. The output layer consists of 1000 neurons corresponding to the number of categories of the Imagenet dataset. As the number of filters increases following the model depth, hence the number of parameters increases significantly in the later layers. Especially, the parameter number in the two fully connected hidden layers is very large, with 102, 764, 544, and 16, 781, 312 parameters, respectively. It accounts for 86.4% parameters of the whole model.

VGG-16 is a pre-trained Deep learning model. A pre-trained model is trained on a different task than the task at hand and provides a good starting point since the features learned on the old task are useful for the new task. Deep Neural networks have a large number of unknown parameters. To find all the unknown parameters would require lots of data (in millions). It is very difficult to get such large labeled dataset. Instead we leverage models that have already being trained on very large amounts ofdata for difficult task with thousands of classes. So instead we use these models as a starting point for our training process, instead of training the own model from scratch.

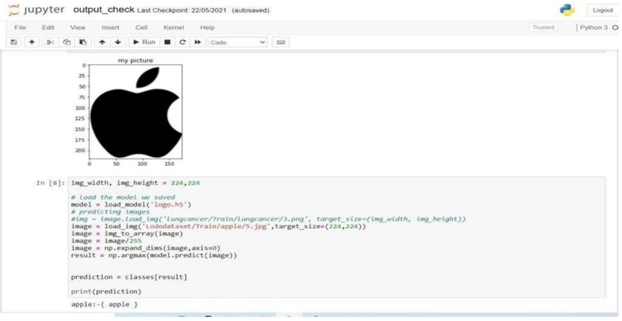
Pre trained weights of ImageNet were loaded from TensorFlow. Then the base layers are frozen to avoid impairment of already learned features. Then new trainable layers are added, and these layers are trained on the collected dataset so that it can determine the features to classify brain hemorrhage.Then the model is fine-tuned, and then the weights are saved. Using pre trained models helps avoid unnecessary computational costs and helps intaking advantage of already biased weights without losing already learned features.

IV. RESULTS

Steps To Find the Result

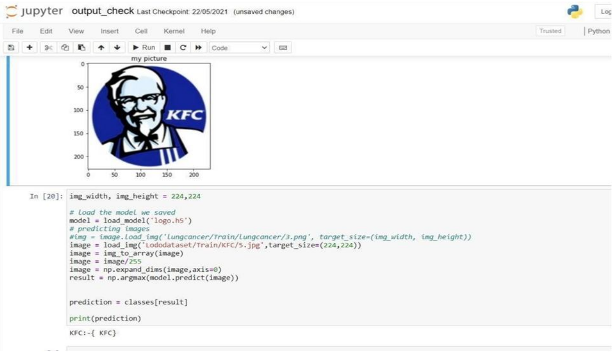
1) Collect Logo images for the desired classes
2) Download the tensorflow modules
3) Run the training code to train the system on all theimages from a desired class
4) Test given set of images for the detection of logos.
5) Use VGG19 module for the purpose of trainingand testing.

A. Screenshots


jupyter output_check Last Checkpoint: 22/05/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
my picture
0
25
50
75
100
125
150
175
200
0
50
100
150

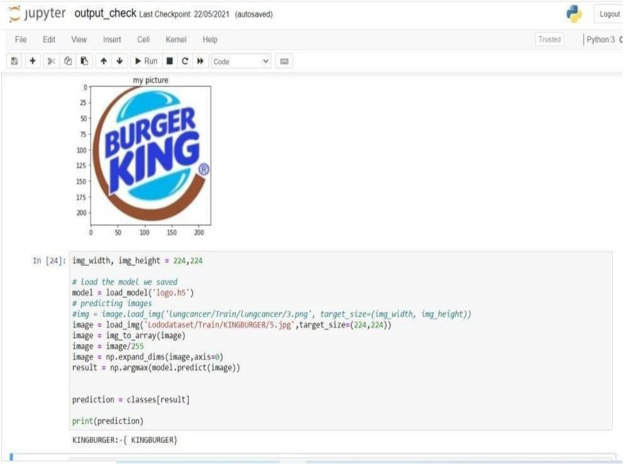
```
In [8]: img_width, img_height = 224,224

        # load the model we saved
        model = load_model('logo.h5')
        # predicting images
        #img = image.load_img('lungcancer/Train/lungcancer/3.png', target_size=(img_width, img_height))
        image = load_img('Lododataset/Train/apple/5.jpg',target_size=(224,224))
        image = img_to_array(image)
        image = image/255
        image = np.expand_dims(image,axis=0)
        result = np.argmax(model.predict(image))


        prediction = classes[result]

        print(prediction)

        apple:-{ apple }
```

Fig 4.1 Prediction of Apple Logo


jupyter output_check Last Checkpoint: 22/05/2021 (unsaved changes)
Log
File Edit View Insert Cell Kernel Help
Trusted
Python
Run
Code
my picture
KFC
0
50
100
150
200
0
50
100
150
200

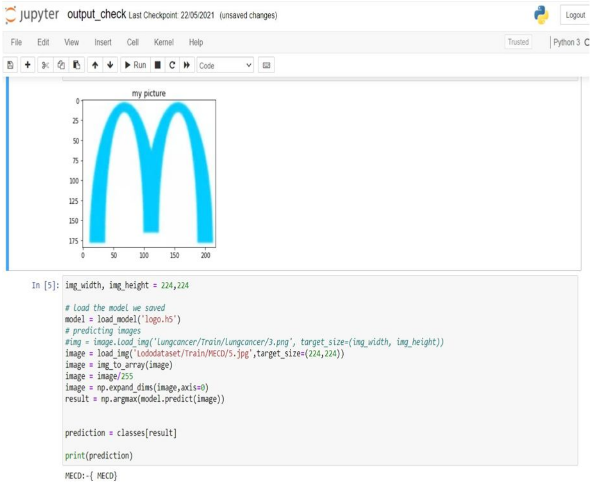
```
In [20]: img_width, img_height = 224,224

         # load the model we saved
         model = load_model('logo.h5')
         # predicting images
         #img = image.load_img('lungcancer/Train/lungcancer/3.png', target_size=(img_width, img_height))
         image = load_img('Lododataset/Train/KFC/5.jpg',target_size=(224,224))
         image = img_to_array(image)
         image = image/255
         image = np.expand_dims(image,axis=0)
         result = np.argmax(model.predict(image))


         prediction = classes[result]

         print(prediction)

         KFC:-{ KFC}
```

Fig 4.2 Prediction of KFC Logo


jupyter output_check Last Checkpoint: 22/05/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
my picture
0
25
50
75
100
125
150
175
200
0
50
100
150
200

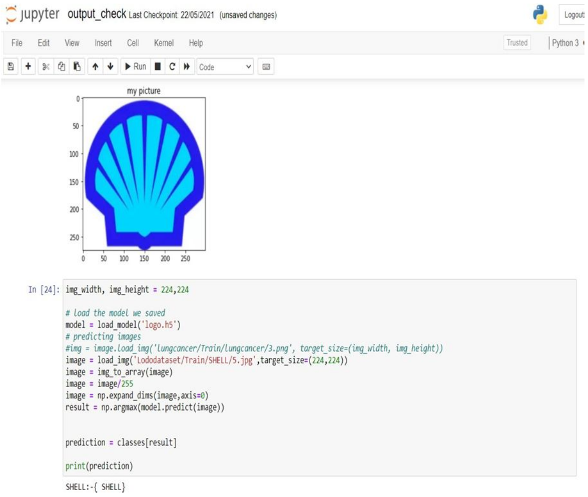
```
In [24]: img_width, img_height = 224,224

         # load the model we saved
         model = load_model('logo.h5')
         # predicting images
         #img = image.load_img('lungcancer/Train/lungcancer/3.png', target_size=(img_width, img_height))
         image = load_img('Lododataset/Train/KINGBURGER/5.jpg',target_size=(224,224))
         image = img_to_array(image)
         image = image/255
         image = np.expand_dims(image,axis=0)
         result = np.argmax(model.predict(image))


         prediction = classes[result]

         print(prediction)

         KINGBURGER:-{ KINGBURGER}
```

Fig 4.3 Prediction of King Burger Logo


jupyter output_check Last Checkpoint: 22/05/2021 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
my picture
0
25
50
75
100
125
150
175
0
50
100
150
200

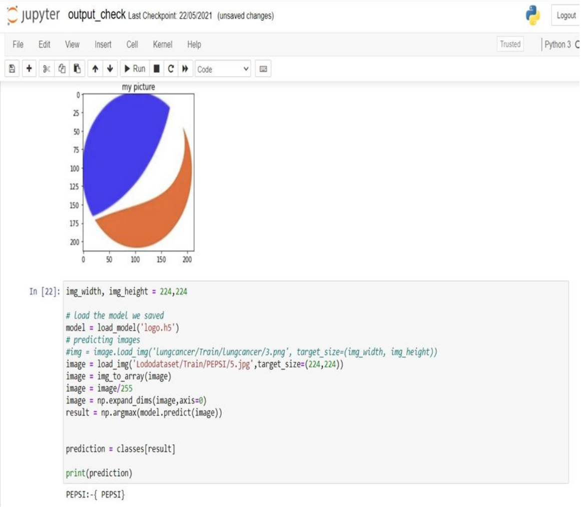
```
In [5]: img_width, img_height = 224,224

        # load the model we saved
        model = load_model('logo.h5')
        # predicting images
        #img = image.load_img('lungcancer/Train/lungcancer/3.png', target_size=(img_width, img_height))
        image = load_img('Lododataset/Train/MECD/5.jpg',target_size=(224,224))
        image = img_to_array(image)
        image = image/255
        image = np.expand_dims(image,axis=0)
        result = np.argmax(model.predict(image))


        prediction = classes[result]

        print(prediction)

        MECD:-{ MECD}
```

Fig 4.4 Prediction of McDonalds Logo


jupyter output_check Last Checkpoint: 22/05/2021 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
my picture
0
50
100
150
200
250
0
50
100
150
200
250

```
In [24]: img_width, img_height = 224,224

         # load the model we saved
         model = load_model('logo.h5')
         # predicting images
         #img = image.load_img('lungcancer/Train/lungcancer/3.png', target_size=(img_width, img_height))
         image = load_img('Lododataset/Train/SHELL/5.jpg',target_size=(224,224))
         image = img_to_array(image)
         image = image/255
         image = np.expand_dims(image,axis=0)
         result = np.argmax(model.predict(image))


         prediction = classes[result]

         print(prediction)

         SHELL:-{ SHELL}
```

Fig 4.5 Prediction of Shell Logo



```
In [22]: img_width, img_height = 224,224

         # load the model we saved
         model = load_model('logo.h5')
         # predicting images
         #img = image.load_img('lungcancer/Train/lungcancer/3.png', target_size=(img_width, img_height))
         image = load_img('Lododataset/Train/PEPSI/5.jpg',target_size=(224,224))
         image = img_to_array(image)
         image = image/255
         image = np.expand_dims(image,axis=0)
         result = np.argmax(model.predict(image))


         prediction = classes[result]

         print(prediction)

         PEPSI:-{ PEPSI}
```

Fig 4.6 Prediction of Pepsi Logo

B. Training a Model

To train a model, we use the VGG19 trained model as the checkpoint to perform transfer learning.

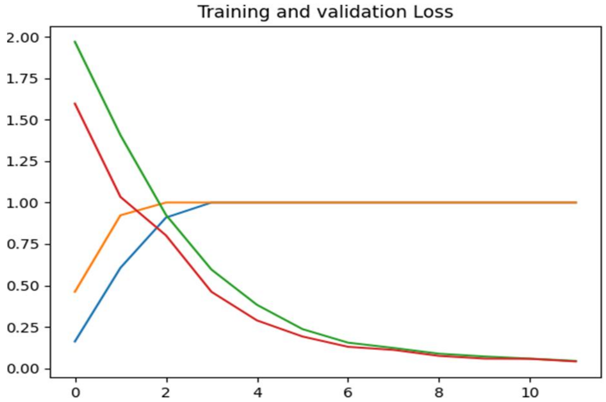


Fig 4.8 Loss value per Epoch graph

C. Validating the Model

Validating the model is nothing but calculating the performance of the model for all possible test cases and scenarios. It also includes comparison of the model with all pre- existing models and their performances

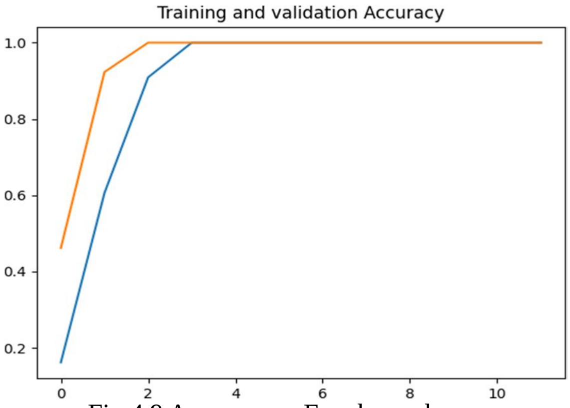


Fig 4.9 Accuracy per Epoch graph

V. CONCLUSION

A key contribution of this work has been the introduction of graphic logo detection using regions‘proposals’ and CNNs, by taking advantage of transfer learning. We experimented with a modern detection model and two CNNs pre-trained for a general object recognition task with abundant data and fine-tuned these networks for a task where the data is scarce - graphic logo detection. Both proposed models perform better than those of the state-of-the-art performance almost out of the box, with a wide margin for improvement that we intend to explore further. In the future, we will extend this model by exploring specific ways to generate regions where the presence of a graphic logo is probable, instead of using a class agnostic method like Selective Search. We are also planning on exploring data augmentation techniques to generaterealistic transformations of graphic logos. Then we can understand if the logos retrievedare the best ones or not, and if the ranking technique used works well.

REFERENCES

[1] Chao Lu, Dandan Li, Dan Zeng, “Logo Detection Based on Convolutional Neural Networks”, 2019 IEEE International Conference onSignal, Information and Data Processing (ICSIDP).

[2] Daniel Mas Montserrat, Qian Lin, Edward J. Delp, Jan P. Allebach, “Scalable Logo Detection and Recognition with Minimal Labeling”, 2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR).

[3] Yuanyuan Li , Qiuyue Shi , Jiangfan Deng , Fei Su ” Graphic Logo Detection with Deep Region- based Convolutional Networks”,2018 IEEE Visual Communications and Image Processing (VCIP).

[4] Goncalo Oliveira, Xavier Frazao, Andre Pimentel , Bernardete Ribeiro 2016, “Automatic Graphic Logo Detection via Fast Region-based Convolutional Networks”, 2018 International Joint Conference on Neural Networks (IJCNN).

[5] Mohammad Wahyudi Nafi,Eko MulyantoYuniarno,Achmad Affandi,” Vehicle Brands andTypes Detection Using Mask R-CNN”,2017International Seminar on Intelligent Technologyand Its Applications (ISITIA).

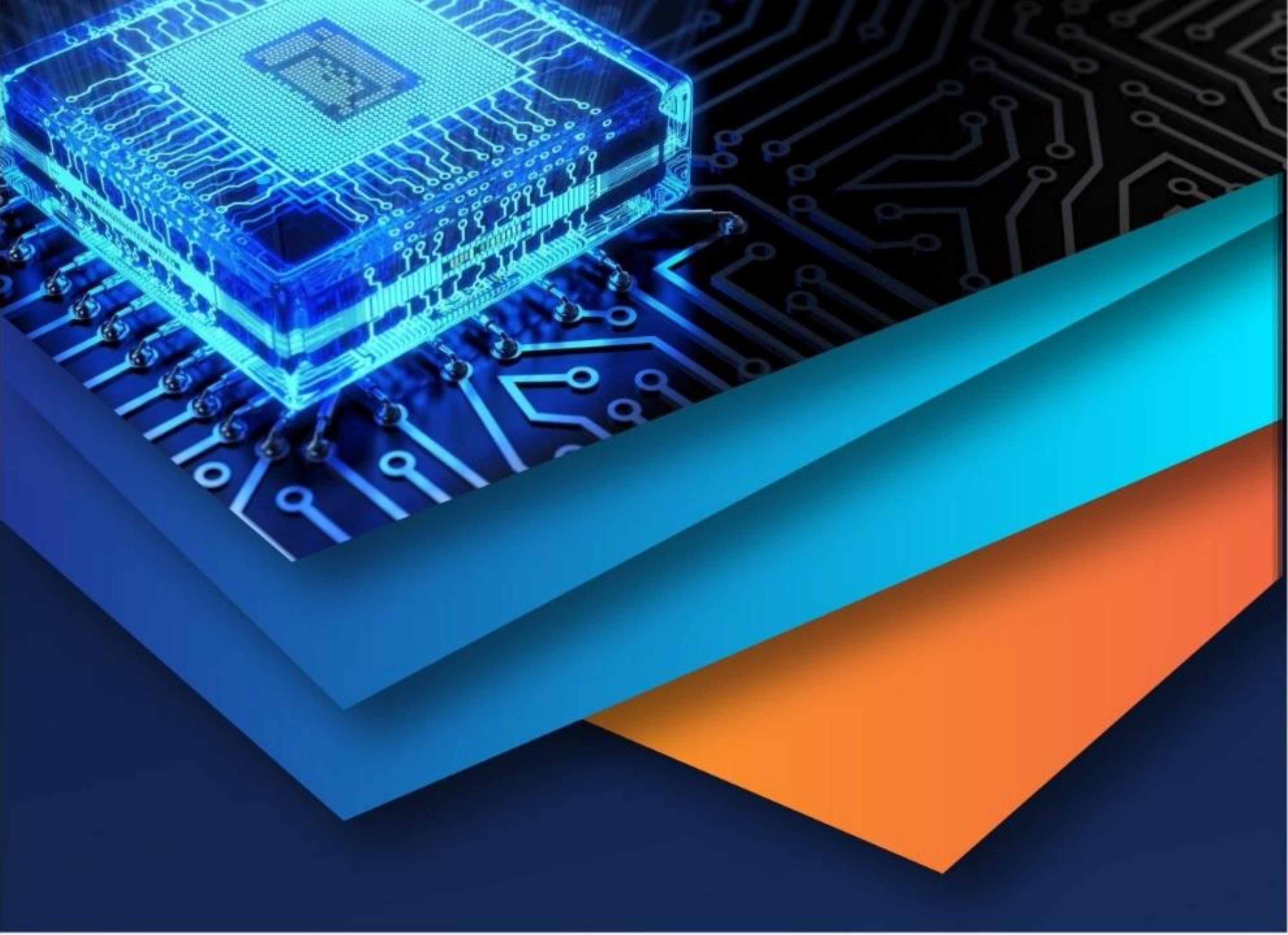
[6] Sarwo,YayaHeryadi,Edy Abdulrachman,WidodoBudiharto,” Logo detection and brand recognitionwith one-stage logo detection framework andsimplified resnet50 backbone”, 2017 InternationalCongress on Applied Information Technology(AIT).

[7] Nhat-Duy Nguyen,Thua Nguyen,Tien Do,Thanh Duc Ngo;Duy-Dinh Le,U15-Logos: “Unconstrained Logo Dataset with Evaluation by Deep learning Methods”,2016 International Conference on Multimedia Analysis and Pattern Recognition (MAPR).

[8] Karel Palecek, “Deep Learning for Logo Detection” 2015 International Conference on Telecommunications and Signal Processing (TSP).

[9] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei,and S. Savarese, “Social LSTM: human trajectory prediction in crowded spaces,” in Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition, pp. 961–971, Las Vegas, NV, USA, June 2016.

[10] P. Pham, D. Nguyen, T. Do, T. D. Ngo, and D.-D. Le, “Evaluation of deep models for real-time small object detection,” in Proceedings of the International Conference on Neural Information Processing, pp. 516–526, Springer, Guangzhou, China, November 2017.



doi crossref

10.22214/IJRASET

INDEX COPERNICUS

45.98

ISRA JIF

IMPACT FACTOR: 7.129

TOGETHER WE REACH THE GOAL

IMPACT FACTOR: 7.429

ISI

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY