ISSN 2278-3091
Volume 8, No.5, September - October 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse100852019.pdf
https://doi.org/10.30534/ijatcse/2019/100852019

Logo Recognition Using Deep Learning and Storing Screen Time in MongoDB Database

Aashreen Raorane[1], Sakshi Patil[1], Lakshmi Kurup[2]
[1]Undergrad Student, Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai 400056, India
[2]Assistant Professor, Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai 400056, India

ABSTRACT

A Logo is the medium through which a company reaches its target audience. Huge amount of funds is invested by companies for every second of screen time of their logo. This project proposes to detect three logos: Adidas, Coca Cola and DHL from various media covered events such as sports or entertainment and store their screen time. A custom dataset containing 1025 images was constructed from sports and entertainment videos. Each image was annotated to its specific logo class. Image preprocessing is done before the images are fed into the Deep Learning model. The model uses Single Shot Multibox Detector (SSD) algorithm along with Inception v2 as a base network to construct the model via Transfer Learning. After training the model, it is used to detect logos in sports videos and the detected logos are stored in a MongoDB database along with their screen time and frequency of occurrence.

Key words: Deep Learning, Image Preprocessing, Inception v2, SSD, Transfer Learning

1. INTRODUCTION

Detection of brand logos has numerous applications that include traffic control systems that detect logo of a car for smart traffic control [1], sports for brand publicity [2], social media for brand data gathering [3] and copyright infringement detection [2].
Publicity for business can be extremely valuable in building credibility and awareness for your company. Many companies use sports endorsements as their key in reaching large number of audiences. In these cases, the screen time of a particular brand logo plays a very important role. The objective of this project is calculating and storing screen time of various brand logos that enables the companies to make significant decisions regarding its investment in publicity.This project can be used by many companies to get an overview on how much its logo reaches the common public and thus guiding the company in important business strategies.The main application of this project lies in satisfying the publicity needs of the company. A company spends huge sums of money on its advertisements and publicity.Thus, it is necessary for the company to know if the investment really pays off. This project can be used to determine the screen time and number of occurrences of the company's or organization's logo to analyze how much it creates and impact on the common public.

2. IMAGE DATASET

2.1 Dataset Gathering

In this project we created our own dataset for four logo classes: Adidas, Coca Cola, DHL and a background class for images that contained no logos. All Images were publicly available and downloaded online or taken from sports videos. A few Rugby tournament videos were gathered and each video was divided into frames using OpenCV. Each frame contributed to one image in the dataset. We created and released over 1000 logo images containing 1963 logos. Some images of the dataset are shown in Figure 1.

Figure 1: Custom Dataset

2.2 Data Annotation

In order to configure the model classes, the images in the dataset are annotated to their specific classes for the model to be trained on them. The images having no logos belonged to the background class. For this process of annotation of our dataset we use LabelImg [4]. LabelImg is a graphical tool

used for annotating images.We have used Pascal VOC format to save the annotations produced as XML files. These XML files can also be summarized using approaches given in the paper [13] for further analysis.

2.3 Converting XML files to TFRecords

While working with larger datasets, it is beneficial to convert our data into a binary file format.Thebinary data requires less disk space and time to copy. It also can be read much more effectively from disk. To reduce the training time of our model
and improve the performance of our import pipeline we use Tensorflow Records, also known as TFRecords. TFRecords are Tensorflow's own binary storage format optimized in multiple ways to be to allow it to be recognized by tensorflow. For very large datasets, the advantage of using TFRecords is that only the data which is required at that particular time is loaded from the disk and processed.

2.4 Preprocessing and Optimization

The dataset is split into test and training set. The training set consists of images that are used to train the model. In order to achieve high performance, we perform data preprocessing on the images in the training dataset. The benefits of using these optimization and preprocessing techniques on the model are discussed in the paper [5].

A. **Data Augmentation**: Training examples are augmented in number by generating random shifts of logo regions. Random shifts are generated in the logo regions in order to augment the data. Along with random shifts, the training data is also augmented by generating a number of images that differ in brightness and contrast. This makes the model more robust to accurately localize and classify logos.

B. **Contrast normalization**: Contrast Normalization is done on the training images. This method involves subtraction of the mean and its division from standard deviation, extracted from each image in the training set. The goal is to make the model sturdy to detect logos in varied contrast and brightness.

C. **Image resizing**: All the images in the training set were resized to 300 x 300 pixels resolution. This increases the performance of the model by reducing the processing time making it both fast and easy to train and readily generalizable to larger image inputs.

D. **Object-proposals**: An object proposal algorithm is employed in the pipeline, on training as well as test images. It extracts regions that are most likely to contain a logo. Each object proposal is then annotated as a corresponding logo class if it overlaps with the manually annotated logo region, otherwise it is annotated as a background class.

E. **Background class**: As stated earlier in the paper, a background class is considered together with the logo classes. It consists of images that the object proposal algorithm generates which do not overlap with any logo annotations. Using a background class increases the accuracy of the model by training it to discriminate the logo in the image from the background.

F. **Thresholding**: After the model is trained, the model detects logos within the image with a certain confidence. If the confidence of the logo class does is below the learned threshold, the candidate region is not classified as a logo. A logo is only predicted when its confidence lies above the threshold that has been learning while training. This optimizes the model by reducing the chances of classifying false positives logos. Only the logos that satisfy the threshold are then stored in a MongoDB database to record their occurrence.

3. IMPLEMENTATION

In this paper, we propose a method that uses Single Shot Multibox Detector (SSD) from the Tensorflow APIs for localization and detection of brand logos in images. We used Inception v2 that was pretrained on COCO dataset to further train it to recognize logos with the use of transfer learning.

3.1 Single Shot Multibox Detector (SSD)

SSD is an algorithm for real time object detection wherein the functions such as localization of objects and classification are done in a single forward pass of the neural network. Faster RCNN creates boundary boxes for object classification that runs at 5-17 frames per second with performance of 73.2% mAPas stated in paper [8], which is very less than the requirement for real time video processing. SSD eliminates the need for region proposal network that is used in Faster RCNN, thus achieving the performance over 74.3% mAP at 59 frames per second on COCO dataset [6]. The architecture of SSD is given in Figure 2. It is based on reputable VGG 16 architecture due to its performance and ability to solve problems that involve transfer learning.

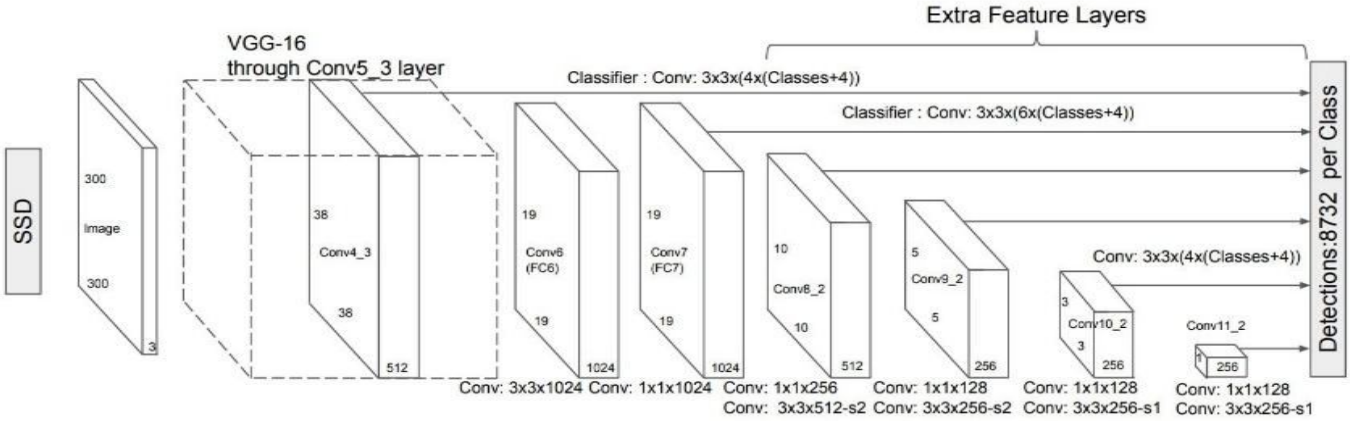

Figure 2:SSD Architecture. Taken from: Liu, Wei et al. "SSD: Single Shot MultiBox Detector." Lecture Notes in Computer Science (2016): 21–37. Crossref. Web

The images are divided into grid cells that are responsible for detecting images. These cells determine the position and shape of the object within the cell. Multiple anchor boxes are assigned to every grid cell. These anchors boxes are pre-defined and of fixed size that closely match the distribution of ground truth boxes. The location and class of the object are predicted with the help of the highest degree of overlap with an object in these boxes.
The loss functions of multibox combines confidence loss and location loss, the two crucial components of SSD. Confidence loss indicates the confidence of network in the computed bounding box around the detected object. It is calculated using categorical cross entropy. Location loss measures the distance between the predicted bounding boxes and the true ones. L2 norm is used for evaluation of location loss. Multibox loss is calculated using Formula (1).

$$Multibox\ loss = confidence\ loss + \propto \times location\ loss \quad (1)$$

In Formula (1), the α term is the balancing factor for location loss.
The results on the standard datasets such as PASCAL VOC, COCO, and ILSVRC confirm that SSD competes for accuracy with the methods that require an additional object proposal step and is much faster. At the same time, it also provides a unified framework for both training and inference.[6]

3.2 Inception v2

The pretrained SSD model is based on Inception v2 and is built in tensorflow. VGGNet has feature of architectural simplicity, but the network evaluation demands lot of computation. On the contrary, the Inception architecture was also designed to perform under strict memory constraints and computational budget. Inception also has lower computational cost as compared to VGGNet or higher successors. The computation cost of Inception v2 is only about 2.5 higher than

that of GoogLeNet despite of it being 42 layers deep. It is still much more efficient than VGGNet[7].

Dimension reduction may cause loss of information, known as representational bottleneck. Inception v2 reduces representational bottleneck by expanding the filter banks. Convolutions are made more efficient computationally using effective factorization methods. Spatial factorization factorizes the convolutions of size nxn to the combination of 1xn and nx1 convolutions on the output. This method is 33% cheaper than a single nxn convolution [7].

3.3 Prediction in Video

For using the model to localize and classify logos in a video, an input video is converted to the image frames using OpenCV. These image frames are then fed one by one to the model for prediction. The model predicts the logos for every frame in the video. The frequency of occurrence as well as the duration of occurrence of every logo is stored in a database. In order to find the duration in seconds for which the logo occurred, the frame rate of the video is determined. Then, the duration in seconds is calculated using the formula (2)

$$duration\ (in\ seconds) = \frac{number\ of\ frames\ for\ which\ logo\ was\ detected\ in\ video}{frame\ rate\ of\ the\ video} \quad (2)$$

3.4 Storing into database

MongoDB is a nonrelational database management system designed for managing data that does not follow rigid relational structure. MongoDB is specifically designed to run on large volumes of unstructured data [9]. In this project, the model is to be run on multiple videos of varying lengths. Since every second in a video consists of several frames and the model is run on each frame, the amount of data collected per video is huge. Further, the screen time of the logo detected

could be used for other analytics of the company at a later time, which is why the schema is not rigid. This makes MongoDB the perfect Database Management System for the project. It is document-oriented and written in C++ [10]. MongoDB consists of set of collections that don't have predefined schema, but the data is stored in binary encoded JSON objects. These collections are capable of containing any type of documents such as images [11]. After detection and classification, the logos whose confidence threshold is satisfied are stored in a MongoDB database along with their screen time and frequency. The contents of the database can then be used for reference for further tasks. We construct a new database known as "logo" to store occurrences of all the logo classes. The collection named as "brand" is made within the database. The database schema followed in this project is:

Database: logo
Collection: brand
{
  _id: ObjectId
  name: "Company Name"
  description: "Company Description"
curr_video: [
  {
duration: "Duration for which logo was displayed in current video (secs)"
frequency: "Number of times logo was displayed in current video"
  }
  ]
total_duration: "Duration for which a brand was displayed in all videos till date"
number_of_video: "Number of videos in which the brand logo occurred"
}

The MongoDB database is used to store the number of times a particular logo occurs and the onscreen time of a particular logo in a particular video. Further, when multiple videos are fed to the model, the database stores the total duration for which a logo was displayed in all the videos fed till date into the model under the variable "total_duration". It also stores the number of videos in which that particular logo was encountered. The complete working of the system has been represented in Figure 3.

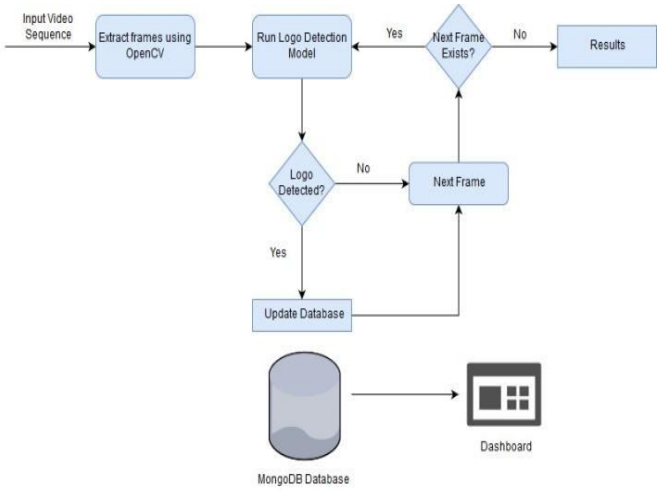

Figure 3: Working Diagram of the system

4. RESULTS

The model was implemented with the given architecture and various optimization steps techniques were employed to increase the accuracy of the model. After training the model a MongoDB database was created to store all occurrences of logos. The model was able to detect the three logos successfully and stored their occurrences in the MongoDB database. The database was designed to self-update every time a new video was run on the model. Figure 4 shows some of the logo recognition results.

Figure 4: Logo Recognition Results

. 5. CONCLUSION

Inferencing, our attempt to recognise and classify brand logos with the proposed architecture and store ther screen time of a brand logo was successful. This screen time and frequency of the logo stored in MongoDB database can be used for data analytics by the company or advertising agencies in order to design specific marketing and business strategies based on the reachability of its logo. Further, the system can also be used to detect copyright infringement which is a pressing problem in the technologically advancing world where huge amounts of data is being created every second which makes it impossible to resolve the issue of copyright infringement manually.

REFERENCES

1. A. P. Psyllos, C.-N. E. Anagnostopoulos, E. 9Kayafas, **Vehicle logo recognition using a sift-based enhanced matching scheme**, Intelligent Transportation Systems, IEEE Transactions on 11 (2) (2010) 322{328. https://doi.org/10.1109/TITS.2010.2042714
2. A. D. Bagdanov, L. Ballan, M. Bertini, A. Del Bimbo, **Trademark matching and retrieval in sports video databases**, in: Proceedings of the international workshop on Workshop on multimedia information retrieval, ACM, 2007, pp. 79–86. https://doi.org/10.1145/1290082.1290096
3. Y. Gao, F. Wang, H. Luan, T.-S. Chua**, Brand data gathering from live social media streams**, in: Proceedings of International Conference on Multimedia Retrieval, ACM, 2014, p. 169. https://doi.org/10.1145/2578726.2578748
4. Tzutalin. LabelImg. Git code (2015). https://github.com/tzutalin/labelImg
5. Bianco, Simone, Marco Buzzelli, Davide Mazzini, Raimondo Schettini: **Deep Learning for Logo Recognition.** Neurocomputing 245 (2017): 23–30. Crossref. Web.
6. Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, "**SSD: Single Shot MultiBox Detector.**" Lecture Notes in Computer Science (2016): 21–37. Crossref. Web. https://doi.org/10.1007/978-3-319-46448-0_2
7. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens and Zbigniew Wojna, **Rethinking the Inception Architecture for Computer Vision**: arXiv:1512.00567 (2015). https://doi.org/10.1109/CVPR.2016.308
8. Shaoqing Ren,Kaiming He, Ross Girshick and Jian Sun, Faster R-CNN: **Towards Real-Time Object Detection with Region Proposal Networks**: arXiv: 1506.01497v3 (2015).
9. Zachary Parker, Scott Poe, and Susan V. Vrbsky. 2013. **Comparing NoSQL MongoDB to an SQL DB**. In Proceedings of the 51st ACM Southeast Conference (ACMSE '13). ACM, New York, NY, USA, Article 5, 6 pages. DOI: https://doi.org/10.1145/2498328.2500047
10. K. Chodorow and M. Dirolf, **MongoDB: The Definitive Guide**. O'Reilly Media, September 2010.
11. K. Sanobar, M. Vanita, "**SQL Support over MongoDB using Metadata**",International Journal ofScientific and Research Publications, Volume 3, Issue 10, October 2013.
12. O. R. Team (2011**) Big data now: current perspectives from OReilly Radar.** OReilly Media.
13. Hassan A. Elmadany, Marco Alfonse, Mostafa Aref, "**A Semantic Framework for Summarizing XML Documents**", International Journal of Advanced Trends in Computer Science and Engineering, Volume 6, No 4, July-August 2017.