Text and Image Query System for Image Datasets

Kejitan Dontas*, Krishna Kumar Tiwari**

* Independent Consultant
** Founder MLAi Community

***Abstract*-** In this paper, we have described the approach we used to build an end to end system for text and image query on image data sets, we also call it TIQS (Text & Image Query System). The system retrieves relevant images from a (given annotated) data set based on a sample image or a text query.
We have used images from two data sets. 1. Visual Genome dataset from Stanford [1][2] and 2. ADE20K data set from CSAIL MIT [3][4][5]. The Visual Genome data set is fully annotated and can be directly used for text queries. To respond to image-based queries, the images are segmented and annotated (into 150 categories) using PSPNET[8]. When a sample image is presented to retrieve similar images, the sample image too is segmented and annotated using PSPNET. A novel similarity score is defined between the sample image and images in the database, and the images with high similarity scores are presented to the user.

***Index Terms*-** ADE20K, Annotated Images, Image Retrieval, Semantic Segmentation, Visual Genome

I. INTRODUCTION

There has been great progress in perceptual tasks such as image classification, object detection etc. In this project, propose a variant approach to search/retrieve desirable images from a suitably annotated image database, given a textual query or a representative image. The underlying model identifies the objects in an image. The query to search/retrieve images typically contains a few objects. The application retrieves the images from a superset of Visual Genome and ADE20K data sets that match the query to a satisfactory extent.

II. TIQS

Problem Specification

TIQS application consists of 1. Text based Query or Text Query 2. Sample Image based Query or Image Query. Text Query uses 108077 images from Visual Genome dataset from Stanford. Image Query uses Visual Genome as well as the 22000 images ADE20K data set from CSAIL MIT.

1. Text Query: The goal is to retrieve images from the superset of Visual Genome Data set such that the retrieved images contain all the objects specified in the Text Input of the Query. Sample input and output (up to 4 images) of this is shown in Figure 3 (Response time < 2 seconds). Images are annotated based on the objects present in the images and these objects are used as indexes as the Elastic Search indexes [7] for fast retrieval of image ids. The annotation files are already present for all 108077 Visual Genome data set images (Fig 1 and 2). These annotations are inserted in the Elastic Search indexes after pre-processing to facilitate query-based image search.

Figure 1. Resized 473x473 pixels image file 1064.jpg from Visual Genome data set

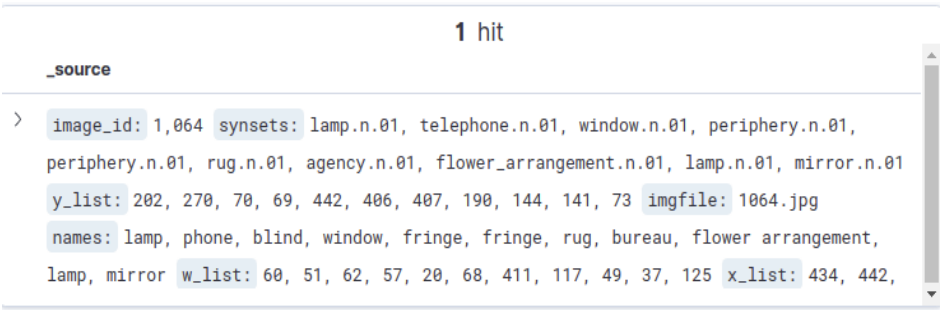

Figure 2. Text annotation in stored in Kibana (Elastic Search)

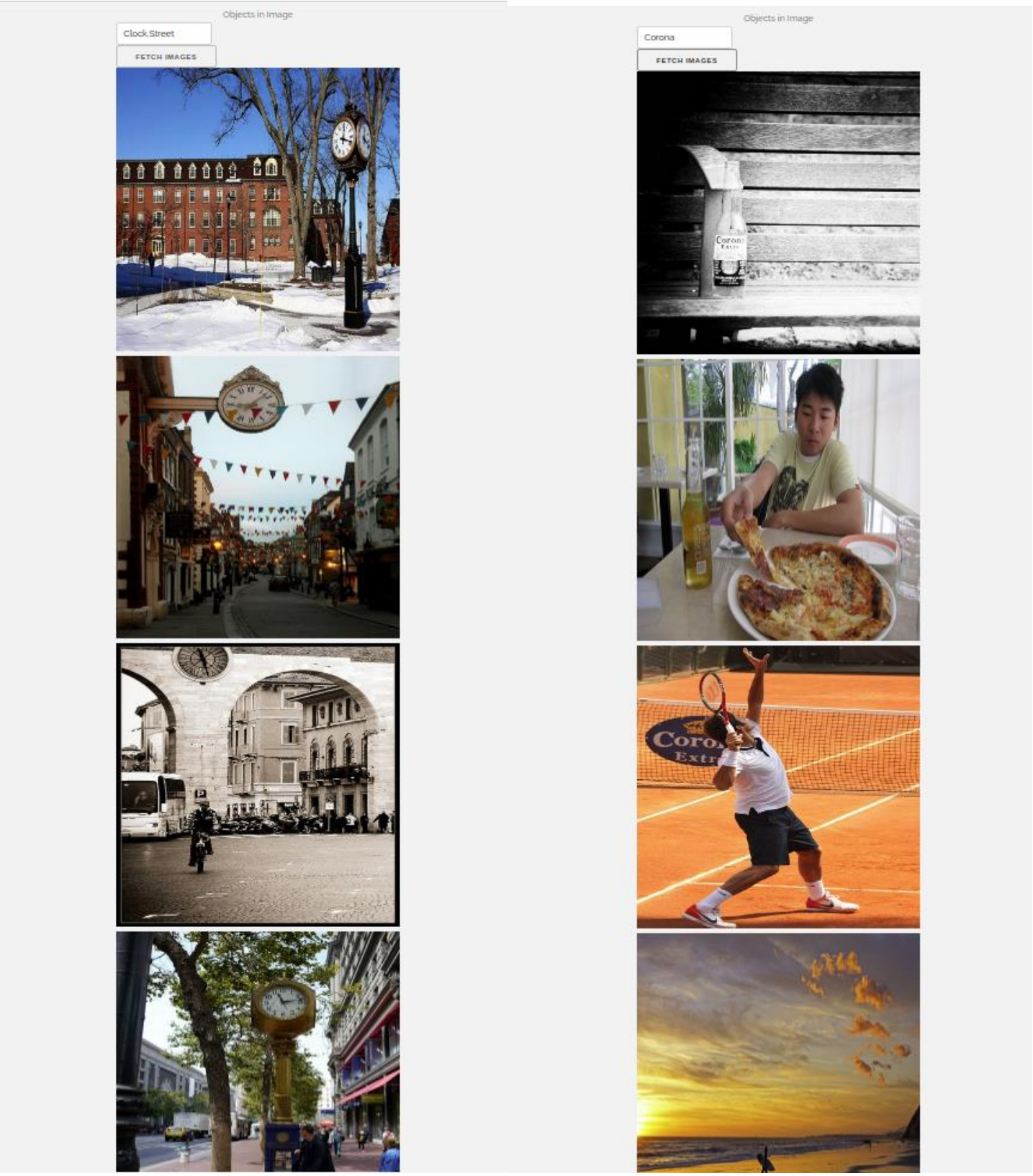

Figure 3: a) Output for query ‘clock, street’ b) output for query ‘Corona’ (beer brand, and astronomical phenomenon)

2. **Image Query**: The goal is to retrieve images from the superset of Visual Genome Data set and ADE20K dataset such that the images are similar to the Sample Input Image. Exemplary output (up to 4 images- configurable) of this query is shown in Figure 4. Response time about 10 seconds with GeForce 1080Ti GPU

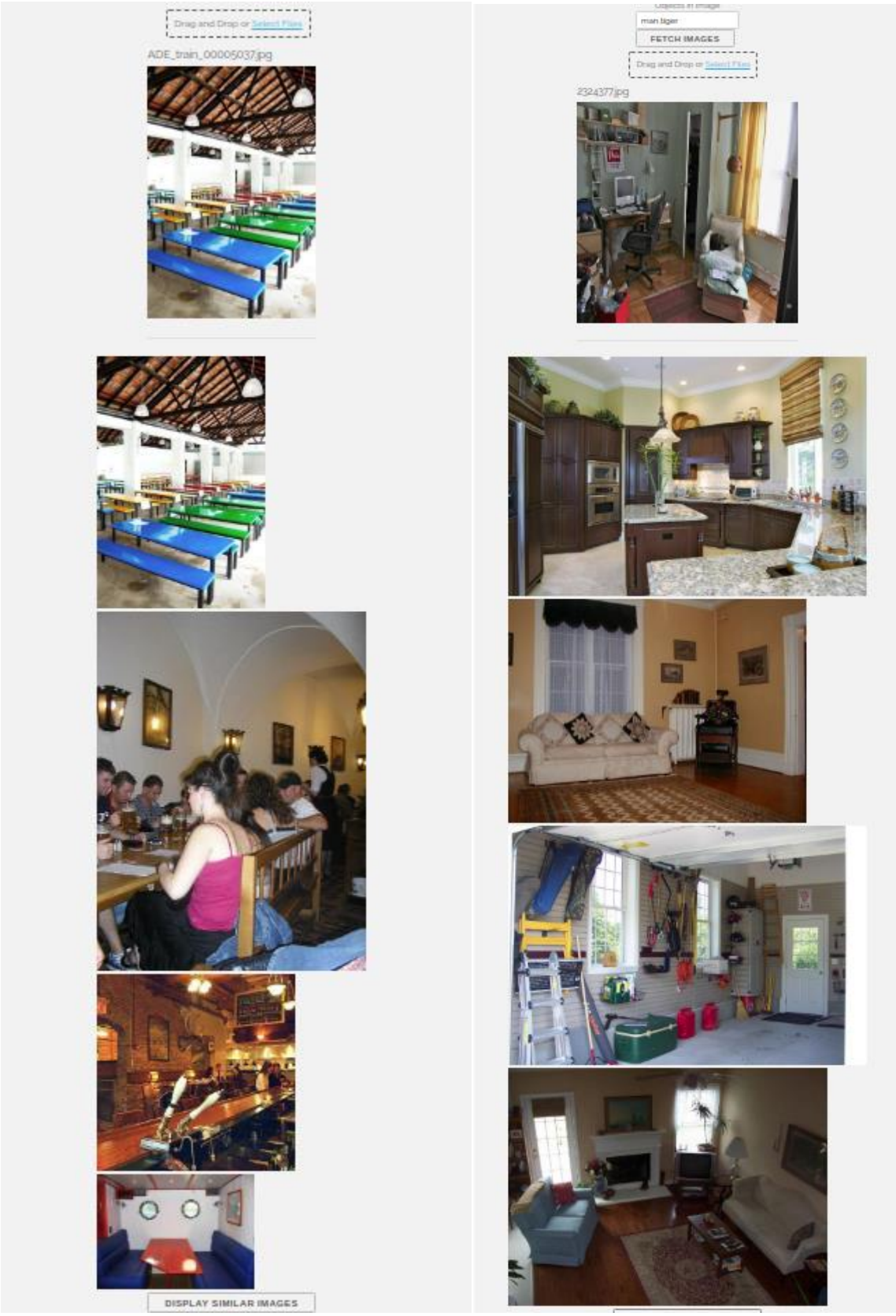

Figure 4. The top most images are Sample input images. The following 4 images are similar images retrieved by the system.

III. OUR APPROACH

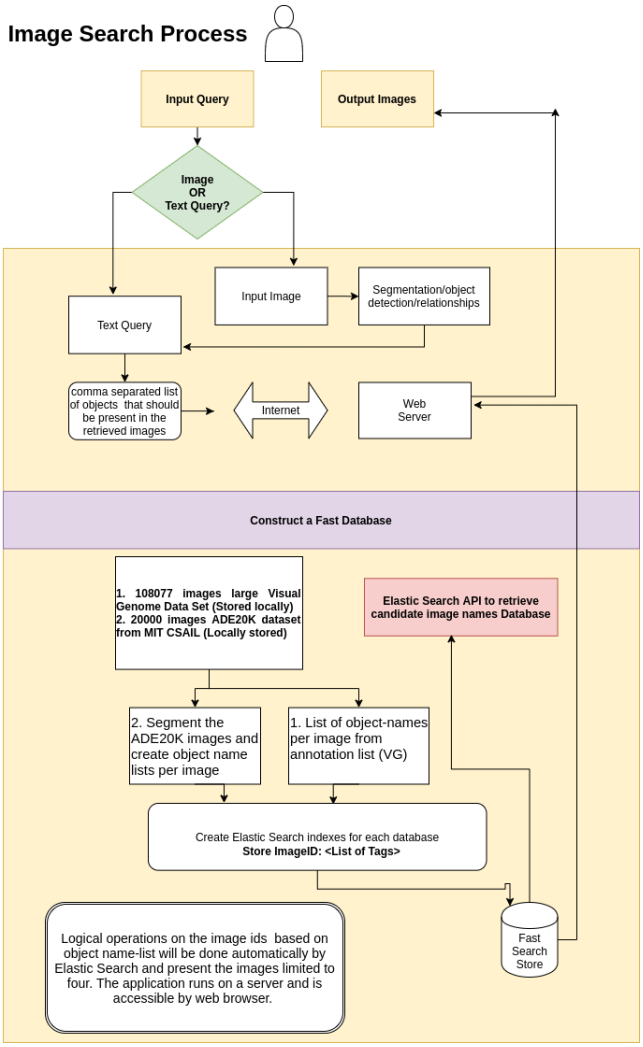

Figure 5. System Diagram

.

Image Query:

1.A sample image is supplied to the application by selecting a sample image file (SampleImg) from local computer or by dragging it on the Drag and Drop of Select Files component.

2. The image is segmented and annotated using [10] that uses PSPNET (Figure 6), and top 11 classes (objects) are retrieved.

3. The number of classes (type of objects in the images) is 150. These classes can be examined in the file objectinfo150.csv (Figure 7).

4. We have 108077 Visual Genome data and 22000 ADE20K images has up to 11 classes identified in the annotations (Figure 8).

5. For similarity, we retrieve image ids of all images that match top 6 classes of the SampleImg. In general, this retrieves a large number of images. Call this set of images as candidates (Cands). We rank the images in Cands. For each Cand in Cands, we compute the set of matching classes MCs. For each C in MCs, we define the similarity contribution SimCls of the class C as

$$\textbf{SimCls} = \frac{PixSampleC}{(abs(PixSampleC - PixCandC) + 10}$$

where PixSampleC = number_of_pixels_assigned_the_class_C in SampleImg, and PixCandC = number_of_pixels_in_the_class_for image Cand. The overall similarity between Cand and SampleImg (SimCand) is defined as the sum of top 6 SimCls.

The formula is chosen so that high similarity score is given if the number of pixels in a class C for Cand is close to that in SampleImg. However, an normalizing factor of 10 is placed in the denominator to give less SimCls to classes/objects with small number of pixels. If the number of pixels for a class C in the Cand and SampleImg differ greatly then the denominator increases and the similarity contribution for the class becomes low. The similarity contribution of all the 6 top matching classes are added and then all the images in the CIDs are ranked and the top 4 are presented to the user.

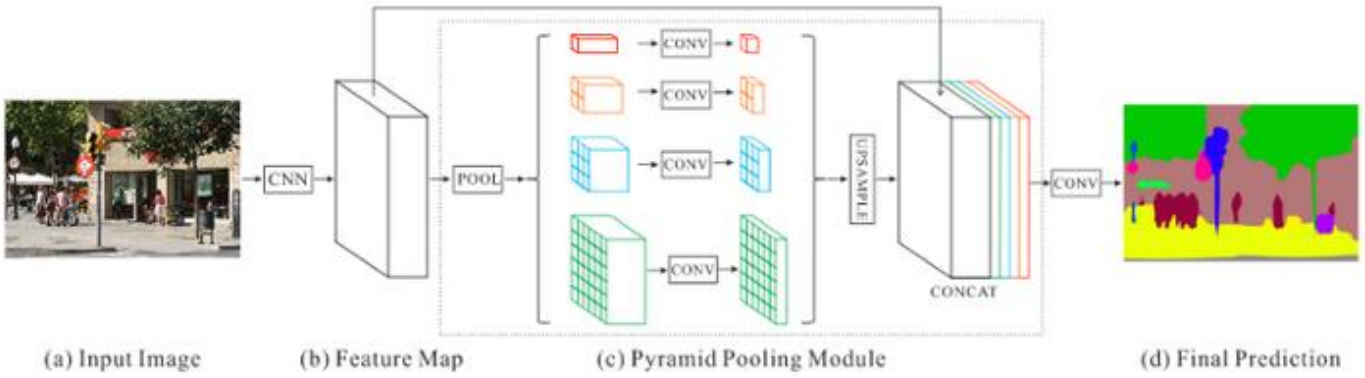

Figure 6. Architecture of PSPNET [9]

Idx	Ratio	Train	Val	Stuff	Name
1	0.1576	11664	1172	1	wall
2	0.1072	6046	612	1	building;edifice
3	0.0878	8265	796	1	sky
4	0.0621	9336	917	1	floor;flooring
5	0.048	6678	641	0	tree
6	0.045	6604	643	1	ceiling
7	0.0398	4023	408	1	road;route
8	0.0231	1906	199	0	bed
9	0.0198	4688	460	0	windowpane;window
10	0.0183	2423	225	1	grass
11	0.0181	2874	294	0	cabinet
12	0.0166	3068	310	1	sidewalk;pavement
13	0.016	5075	526	0	person;individual;someone;somebody;mortal;soul
14	0.0151	1804	190	1	earth;ground
15	0.0118	6666	796	0	door;double;door
16	0.011	4269	411	0	table
17	0.0109	1691	160	1	mountain;mount
18	0.0104	3999	441	0	plant;flora;plant;life
19	0.0104	2149	217	0	curtain;drape;drapery;mantle;pall
20	0.0103	3261	318	0	chair
21	0.0098	3164	306	0	car;auto;automobile;machine;motorcar
22	0.0074	709	75	1	water
23	0.0067	3296	315	0	painting;picture
24	0.0065	1191	106	0	sofa;couch;lounge
25	0.0061	1516	162	0	shelf
26	0.006	667	69	1	house
27	0.0053	651	57	1	sea
28	0.0052	1847	224	0	mirror
29	0.0046	1158	128	1	rug;carpet;carpeting
30	0.0044	480	44	1	field
31	0.0044	1172	98	0	armchair

classnum	classval	xmin	xmax	ymin	ymax	boxarea	density	centroidx	centroidy	classname
1	99682	0	472	0	472	222784	0.4474	224	147	wall
8	70791	32	443	193	472	114669	0.6174	225	336	bed
4	14621	41	450	408	472	26176	0.5586	257	452	floor;flooring
9	9998	134	286	69	251	27664	0.3614	255	158	windowpane;window
16	8122	389	452	282	466	11592	0.7007	419	364	table
45	5840	54	156	196	269	7446	0.7843	102	227	chest;of;drawers;chest;bureau;dresser
19	5073	125	297	23	127	17888	0.2836	241	48	curtain;drape;drapery;mantle;pall
37	3941	118	456	131	295	55432	0.0711	320	213	lamp
23	3927	309	464	56	183	19685	0.1995	364	126	painting;picture
6	679	24	309	0	5	1425	0.4765	208	1	ceiling
67	476	67	94	151	179	756	0.6296	78	165	flower
28	392	79	149	100	146	3220	0.1217	129	119	mirror
136	187	79	89	178	197	190	0.9842	84	187	vase

Figure 7. Partial listing of file objectinfo150.csv indicating class numbers and class names (object names), and some more fields
Figure 8 Annotation file generated from PSPNET. File 1064.txt represents annotation of image file 1064.jpg/1064.png from Visual Genome data set. ClassVal represents the number of pixels for that class in the image. There are some more fields too.

IV. TIQS Implementation and Deployment

Please refer to https://github.com/kejitan/ESVGscale for the details of the implementation and execution
The application was deployed on AWS and could be accessed from a web browser till about 23rd March 2020. Shortly the application will be hosted on Paperspace.

Text Query System

Visual Genome image dataset

1. Download Version 1.2 from https://visualgenome.org/api/v0/api_home.html
2. Convert the .jpg images to (473x473) size png images
3. There are 10 annotation files on the Visual Genome server. We use image_data.json and objects.json files. The annotations indicate image id, image locations, names of objects in the images.
4. Use ESMAP.py to create indexes IDX0 and IDX1 in the Elastic Search system for fast image label data retrieval.

5. Steps 1-4 are one-time process.
6. In the operational mode, the user presents a comma separated list in an Input Text box of the screen.
7. From the input. the system constructs an Elastic Search Query and returns a list of image labels satisfying the user query. The Application uses Plotly Dash and waitress package for hosting

Image Query System

Visual Genome and ADE20K image datasets

1. Download ADE20K full data set from https://groups.csail.mit.edu/vision/datasets/ADE20K/
2. Download Visual Genome Version 1.2 from https://visualgenome.org/api/v0/api_home.html.
3. Use jpg2png.py to convert jpg images to 473x473 png images
4. Use png2seg.py segment the images.
5. Use seg2ann.py to create annotation files
6. Use ESANNadkScore.py with a different directory name in the program to add to index 'vgnum' in Elastic Search.
7. Steps 1-6 are one-time process.
8. In the operational mode, the user presents a sample image via a file name or by Drag and Drop method.
9. The system constructs an Elastic Search Query and returns a list of image labels satisfying the user query. The Application uses Plotly Dash and waitress package.
10. The system can be trained on additional databases by fine-tuning the pretrained network.
11. There is also an inherent need to retrain the model, if the number of classes changes. For example, unlike the Imagenet database that has 1000 or 10000 categories, PSPNET has been trained on 150 classes. This can go up to max 255 as one of the color channels (uint) in the segmented image is used to indicate class number. But then in semantic segmentation 150 classes is already a high number compared to other semantic segmentation models. Fresh training will also be needed if the number of classes is small or the type of objects in the domain are significantly different.
12. The models are conveniently stored as .h5 files and can be reloaded as and when required.

V. CONCLUSION

In Google image search it is possible to present a query in text form or provide a sample image and get relevant images from the web. However, the images are searched from the web and the user does not have control over what image database should be searched. Using custom image database and training it on custom classes provides a great opportunity to create novel applications.

Semantic segmentation, a key step in object recognition has been studied for a while. Typically, UNet, SegNet, Yolo and other models have been used. Many of these are fast compared to PSPNET however, typically they are trained on a fewer number of classes, thus rendering unsuitable for computing similarity between sample image and the images in the data set.

Extensions

1. If it is desired to retrieve images not only exactly as specified in the query but similar to that then we can construct a Word2Vec or GloVe, and WordNet based system that will facilitate this requirement [6].
2. Elastic Search query can be extended to implement the queries containing objects in AND/ OR sense.
3. We can plan to analyze the distribution of the class names across the whole dataset and rank the class names in the individual image annotations accordingly (in a TF/IDF sense) and then use these high-ranking class names for similarity computation
4. Recently there has been work done on PixelLib by and presented in [12],[13]. The PixelLib uses deeplabv3_xception65_ade20k model.
5. We need to come up with a scheme to train a model with custom set of classes. This is very important since the CSAIL pretrained PSPNET50 network with predefined 150 classes is not suitable for arbitrary applications. We also need to speed up the inference, since it takes 1-2 seconds with GPU for segmentation, this is slow for video based applications.

Applications

1. Retrieve Video frames from mpeg file where a team scores a goal. Here the video can be annotated real-time or offline, with custom class names, and the annotations be stored in Elastic Search engine and desired frames can be retrieved instantaneously by specifying objects that should be contained in the video frames. In these kinds of applications, the classes trained in ADE20K dataset will not be suitable and a new model will need to be trained on custom data with custom classes. The process involves annotating (labeling) the segmented image. This is expensive. An alternative is to use a model with close enough domain and then retrain with transfer learning.
2. Retrieve images where a suspicious person approaches a store. Similar to application in 1.
3. Retrieve photos of a friend when they were a child. Here we are thinking about mobile based application where the image database is present in the user mobile. Some interface should be given to the user so that he can annotate the images, name the persons in the photo, name the locations where the photos were taken, specify time when the photos were taken, occasions, and any other tags that will be helpful. We will not be able to use Elastic Search (maybe it is available for mobiles now??). But the database will be small and normal search over the annotations will not be very slow.
4. Applications 1. And 2 can be implemented on edge devices like Raspberry Pi, Intel Movidius NCS2 with OpenVino etc. Other options are Google Coral, Jetson Nano SDK toolkit and many other low end- high end devices. The pspnet50_ade20k.hy model size is about 188 MB and will not fit into the edge devices. There are ways to circumvent this. OpenVino allows converting the models to Intermediate Representation files and then convert them to leaner models that can be loaded into edge devices such as Intel Movidius NCS2.

Overall, the performance of the system is satisfactory. Text based query presents up to 4 images in less than 2 seconds. The image query outputs the results in about 10 seconds with a GPU on the host system. GPU is beneficial for performance, since segmentation and annotation steps of the sample image are expensive. If less than 6 classes are not common in a candidate image, it is not shortlisted for computation of similarity. Thus, if a very novel image is presented, the system does not present any result.

ACKNOWLEDGMENT.

The authors acknowledge use of Divam Gupta's keras-semantic-segmentation library, Stanford University's Visual Genome dataset, MIT CSAIL's ADE20K dataset, pspnet50 model pretrained on ADE20K dataset. The application used Elastic Search, Kibana, Plotly Dash, AWS Educational Credits for hosting the application for a few days. Google Search and Stack Overflow among others were very helpful in resolving myriads of problems encountered while implementing the system. The authors would like to thank Springboard where the first author was a student of AI-ML Career Track and the second author was his mentor.

REFERENCES

[1] Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li Jia-Li, David Ayman Shamma, Michael Bernstein, Li Fei-Fei Yuanzhi Liang, Yalong Bai, Wei Zhang, Xueming Qian, Li Zhu, and Tao Mei: VrR-VG: Refocusing Visually-Relevant Relationships, ArXiv 1902.00313

[2] https://visualgenome.org/api/v0/api_home.html -- Data download directory

[3] Scene Parsing through ADE20K Dataset. Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso and Antonio Torralba. Computer Vision and Pattern Recognition (CVPR), 2017.

[4] Semantic Understanding of Scenes through ADE20K Dataset. Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso and Antonio Torralba. International Journal on Computer Vision (IJCV).

[5] https://groups.csail.mit.edu/vision/datasets/ADE20K/ -- Data download directory

[6] https://cs224d.stanford.edu/reports/PatelShabaz.pdf

[7] https://www.tutorialspoint.com/elasticsearch/index.htm

[8] Hengshuang Zhao,Jianping Shi, Xiaojuan Qi, Xiaogang Wang, Jiaya Jia,, arXiv:1612.01105. Pyramid Scene Parsing Network

[9] https://divamgupta.com/image-segmentation/2019/06/06/deep-learning-semantic-segmentation-keras.html Blog

[10] https://github.com/divamgupta/image-segmentation-keras Github repository

[11] https://code.google.com/archive/p/word2vec/

[12] https://towardsdatascience.com/semantic-segmentation-of-150-classes-of-objects-with-5-lines-of-code-7f244fa96b6c

[13] https://github.com/ayoolaolafenwa/PixelLib