1. INTRODUCTION

1.1 OVERVIEW

1.1.1 Image Processing

Image processing is a method to perform some operations on an image, to get an enhanced image or to extract some useful information from it. The input is an image and output may be an image or features associated with the image. The two types of methods used for Image Processing are Analog and Digital Image Processing. Image Processing provides a comprehensive set of standard algorithms for image processing, analysis, visualization, and algorithm development. It can perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration, and 3D image processing. It can also segment image data, compare image registration techniques and batch-process large data sets.

In the project, it is done using image processing for segmenting the image data and for dealing with relationships among the objects in the image. Segmentation refers to dividing the image into groups of pixels based on some criteria. Semantic segmentation is used where the goal is to assign label to every object in the image.

1.1.2 Machine Learning

Machine learning is a subject that uses statistical techniques to give computers the ability to "learn" with data, without being explicitly programmed. The basic premise of machine learning is to build algorithms that receives input data in the form of text file, images or audio and use statistical methods to predict an output. Machine learning algorithms are often categorized as supervised or unsupervised.

i. Supervised Algorithms**:** Supervised algorithms can be applied to what has been learned in the past to new data using labelled examples to predict future events. From a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system can predict the output values based on the training dataset. It can also compare its output with the correct class labels already predicted in the training dataset and find errors in order to rectify the model accordingly.

ii. Unsupervised Algorithms: Unsupervised algorithms are used when the information used to train is neither classified nor labelled. Unsupervised algorithms studies how systems can infer a function to define a class label from unlabelled data. The system explores the data and can draw inferences from datasets to describe the unlabelled data. Unsupervised algorithms do not need to be trained with desired outcome data. They are used for more complex processing tasks than supervised learning systems, including image recognition, speech-to-text and natural language generation.

1.1.3 Artificial Neural Network

An Artificial Neuron Network (ANN) is a computational model based on the structure and functions of biological neural networks. Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, based on that input and output. ANNs are considered nonlinear statistical data modelling tools where the complex relationships between inputs and outputs are modelled or patterns are found.

A neural network is a series of algorithms that attempts to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. They help to group unlabelled data according to similarities among the inputs, and they classify data when it has a labelled dataset to train. There are several kinds of artificial neural networks. This type of neural networks are implemented based on the mathematical operations and a set of parameters required to determine the output.

i. Feed forward Neural Network
ii. Radial basis function Neural Network
iii. Kohonen self-organising Neural Network
iv. Recurrent Neural Network
v. Convolutional Neural Network

ANNs have three layers that are interconnected. The first layer consists of input neurons. Those neurons send data on to the second layer, which in turn sends the output neurons to the third layer.

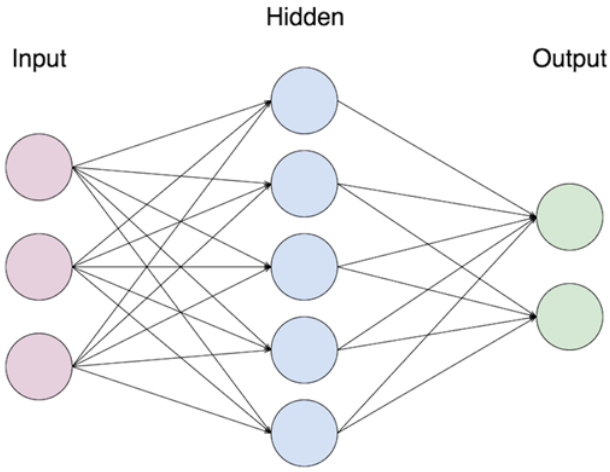

Fig 1.1 Artificial Neural Network

Deep learning is the name used for “stacked neural networks”, which means networks are composed of several layers and each layer consists of nodes. A node is the place where computation happens. It combines input data with a set of coefficients or weights and amplifies the input and signifies which input is most helpful is classifying data without error. The input and weights product are summed up and the sum is passed through a node called activation function, to determine whether and what inputs have to progress further to affect the output.

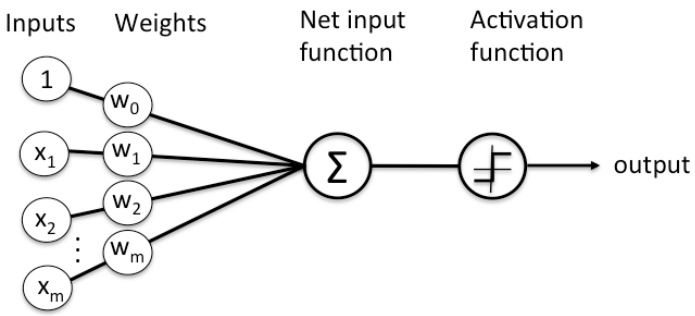

Fig 1.2 Forward Propagation

1.1.4 Convolutional Neural Networks

CNN's, type of neural networks, are made up of neurons with learnable weights and biases. The input features are taken in batch wise like a filter. The network remembers the images in parts and performs computation. These computations involve conversion of the image from RGB or HSI scale to gray-scale. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. There are two components:

i. Hidden Layer/ Feature Extraction Point: In this, the network will perform a series of convolutions and pooling operations during which the features are detected. In convolutional layer, weight matrix behaves like a filter in an image extracting particular information from the original image. When it deals with multiple convolutional layers, the initial layer extract more generic features, while as the network gets deeper, the features extracted are more and more complex. The filter moves across the entire image one pixel at a time. When the weight matrix moves 1 pixel at a time, it is called as a "stride" of 1. As it increases the stride value, the size of the image reduces. The initial shape of the image is retained only after it is padded the image with zero. It periodically introduces pooling layers between subsequent convolutional layers. It is performed to reduce the size of the image. The most common form of pooling layer is the max pooling.

ii. Classification Step: Neurons in a fully connected layer have connections to all the activations in the previous layers. The fully connected layers will serve as a classifier on top of these extracted features. The output is then generated through the output layer and is compared to the output layer for error generation. A loss function is defined to compute the mean square loss. The gradient of error is then calculated. The error is then back-propagated to update the filter and bias values. They will assign a probability for the object in the image based on the algorithm.

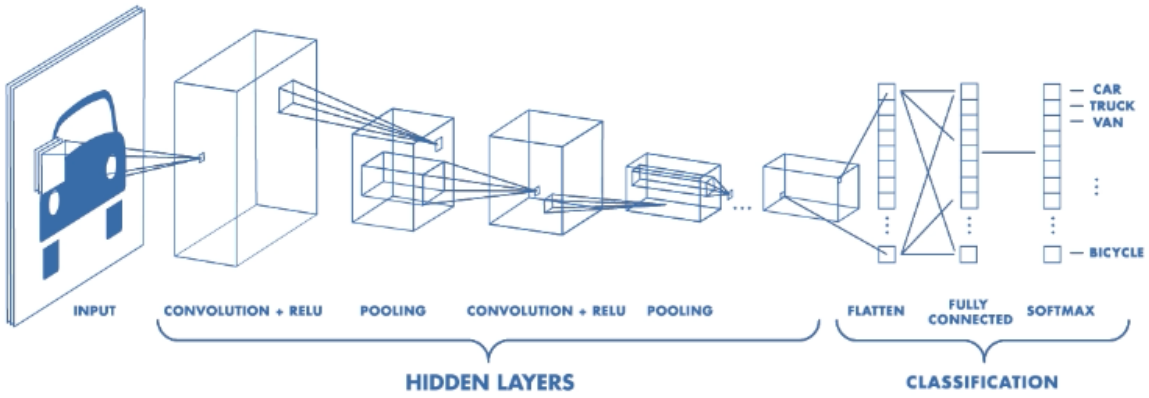

Fig 1.3 Sample CNN

The project deals with a Convolutional Neural Network to classify the images in the Visual Appearance model. Objects are the core building blocks of an image in this model and find a holistic representation. For example, an image with a person and a bicycle will involve pushing, riding and also falling off the bike. This model has a diversity of models which involves accurate image retrieval and semantic understanding of the image. This model extends to give an object recognition of an image and also determines to find the count of objects and the number of segments. Visual Relationship Detection involves detecting and localizing pairs of objects in an image.

1.2 APPLICATIONS OF NEURAL NETWORKS

The following are the applications of Neural Networks:

i. Character Recognition: Handwritten character recognition has been one of the active and challenging research areas in the field of image processing and pattern recognition. It has numerous applications which include, reading aid for the blind, bank cheques and conversion of any handwritten document into structural text form.

ii. Stock Market Prediction: Stock market prediction is the act of trying to determine the future value of company stock or other financial instrument

traded on an exchange. The successful prediction of a stock's future price could yield a significant profit. It analyses the company's future profitability on the basis of its current business environment and financial performance and also includes reading the charts and using statistical figures to identify the trends in the stock market.

iii. Semantic Parsing and Question Answering: Question Answering Systems automatically answer different types of questions asked in natural languages including definition questions, biographical questions, multilingual questions, and so on. Neural networks usage makes it possible to develop high performing question answering systems.

iv. Paraphrase Detection: It determines whether the two sentences have the same explanation. This task is important for question answering systems as there are many ways to ask the same question.

v. Language Generation and Multi-document Summarization: Natural language generation has many applications such as automated writing of reports, generating texts based on analysis of sales data, summarizing electronic medical records, producing textual weather forecasts from weather data, and so on.

vi. Speech Recognition: Speech recognition has many applications, such as home automation, mobile telephony, virtual assistance, hands-free computing, video games, and so on.

vii. Spell Checking: Most text editors let users check if their text contains spelling mistakes. Neural networks are now incorporated into many spell-checking tools.

1.3 ADVANTAGES OF NEURAL NETWORKS

The following are the advantages of Neural Networks:

i. Storage of information: The complete data is stored in a network but not in the database. An absence of few pieces in data will result in malfunction of the neural network.

ii. Ability to work with incomplete information: After the training of the data in the network, it can produce the output of a problem after a few computations.

iii. Fault Tolerance: Corruption of one or more neurons in any layer won't prevent it from generating the output. But the output might be defective.

iv. Non-linear data processing: Nonlinear systems have the capability of finding shortcuts to reach computationally expensive solutions. These systems can also infer connections between data points, rather than waiting for records in a data source to be explicitly linked. This nonlinear short-cut mechanism is fed into artificial neural networking, which makes it valuable in a commercial big-data analysis.

1.4 DISADVANTAGES OF NEURAL NETWORKS

The following are the disadvantages of Neural Networks:

i. Hardware dependence: Artificial neural networks require processors with parallel processing power, in accordance with their structure. For this reason, the realization of the equipment is dependent.

ii. Unexplained behaviour of the network: This is the most important problem of ANN. When ANN produces a probing solution, it does not give a clue as to why and how. This reduces trust in the network.

iii. Determination of proper network structure: There is no specific rule for determining the structure of artificial neural networks. The appropriate network structure is achieved through experience and trial and error.

iv. The duration of the network: The network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum results.

1.5 PROBLEM STATEMENT

Multiclass Object Detection is a task where the input is a large dataset with simple images and the model is supposed to predict a set of relationships that are true about the image. Their relationships are in the form of <object1-predicate-object2>. This is an extension to object detection where it not only detects the objects in an image but also predicts how they are interacting with one another.

Scene Parsing is where the objects in an image are segmented according to the object categories. This is extended to identifying objects with a label, accuracies and count of objects for every image.

1.6 ORGANIZATION OF THE REPORT

The report is organized as follows: Section 2 deals with the Literature Survey which gives an extensive idea about all the previous work that has been carried out in this area. Section 3 focuses on the design of the model and used in the project. Section 4 gives an insight into the implementation, detailing the code snippets and flow of the work. Section 5 deals with the testing and results of the work carried out followed by Section 6 which gives the conclusion and future scope.

2. LITERATURE SURVEY

According to Ruslan Salakhutdinov, Antonio Torralba, Josh Tenenbaum, object localization and detection in a hierarchical manner increases the accuracy of single object detection. It consists of different types of objects which are frequent, less frequent, rare, extremely rare. Using this biased distribution, they use frequent data for classifying rare objects.

Given an image, they distinguish individual objects from the background class to increase the accuracy. The hierarchical model consists of a Global class, Superclass and Class specific. Each node in the tree is represented using parameter,

$$\beta^{(van)} = \theta^{(0)} + \theta_2^{(1)} + \theta_4^{(2)} \qquad \textbf{Eq 2.1}$$

In every separate classification model, they considered a dataset with "N" object classes and training examples. The representation of the vector is (x_i, y_i) where x_i is the training example and y_i is the class label. The class labels are given between {-1,1}. In the binary classification method, each class is treated as a single entity. For each class, the probability and zero mean Gaussian priors are computed.

Learning when a class hierarchy is available:

i. Every node in the model is assigned a parameter vector, vector space for each leaf node is the sum of parameter vectors along the tree.
ii. During the learning phase, a tree structure is built and model the parameters for getting the suitable tree hierarchy.

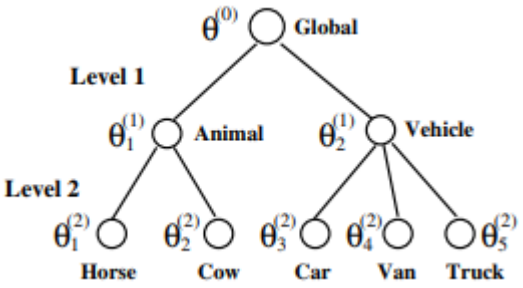

Fig 2.1 Object detection in a hierarchical manner

During the evaluation, the Shared class model is compared with Single class and Global prior. Global prior deals with the parameters **θ**. A single class model does not require a class hierarchy.

For example, the objects like chair, armchair, deckchair are referred under the same category in Shared class model and Global prior. This is because the class-specific labels draw the visual appearance from the intermediate class level as they share the same parameters. Even the performance of infrequent objects is also increased. A confusion matrix is drawn between the shared class model and a single class model.

For detecting cars among a set of training examples, the detector detects car with an average precision of 59.05% in the single class and 59.21% in the shared class. Similar objects such as a truck, van, bus have a low average precision in a single class as visual appearance is not drawn from the higher levels. In a shared class, the average precision for the similar objects to the car is more.

Table 2.1 Average precision for different objects in shared and single class

Single Class Model		Shared Class Model	
Truck	2.88	Truck	10.18
Van	8.19	Van	16.51
Bus	0.87	Bus	4.18

They concluded by determining the increase in the accuracy of single object detectors. This can be used increasing the performance in visual relationship detection.

According to Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning, they focus on dealing with complex image descriptions using rule-based and classifier-based scene graph parser. The main problem with image retrieval systems is that, it represents a particular image according to the complex query. The keyword-based system has failed in this case because the user has to specify deeper semantics of the image. This problem can be solved by using a

Scene Graph which deals with capturing objects, attributes and the relationship between them. The goal of the paper is to parse image descriptions into scene graphs and retrieve images from those scene graphs. Rule-based parser and classifier parser depends on the semantic representation of the sentence.

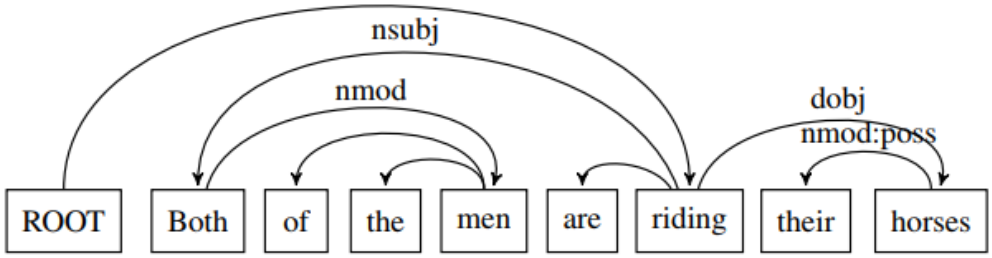

Fig 2.2 Semantic representation of sentences

In the rule-based parser, the scene graph is constructed into adjective modifiers, prepositional phrases, subject-predicate-object structure. The objects, attributes and the relation between them are directly retrieved from the semantic graph. In the classifier parser, initially the objects, attributes are predicted from the semantic graph and then the relation between the objects and the attributes is found. The two parsers which are tested based on two baselines: are Nearest neighbour and Object-only.

After the parser is generated for a sentence, its output is considered as a query. During evaluation, human constructed parser also generates its own output query. From the query generated, a set of images are constructed where all these images are ranked. During the training phase, Classifier parser performs better than Rule parser. Rule-based median ranks 17, Classifier based median ranks 16. During the test phase, both the parsers generate the same results. The change during the training phase is due to the set of images which consists of more annotated images. In comparison to baselines, the Object-only and Classifier parser generates different sets of images on generating the query. For example, given the sentence “The white plane has one blue stripe and one red stripe”, then the Object-only generated images with only two

striped objects without referring to the plane, whereas the Classifier parser generates correct output images containing a plane with two stripes

The drawbacks with these parsers are they lead to errors when dealt with textures, for example, ‘pink dress with polka dots’, ‘A jockey wearing a green helmet and a matching shirt’. The parser cannot infer that the shirt is also green in colour. Another drawback is that the both the parsers depend on the semantic graph for training the data. Representation of incorrect semantic graph leads to wrong results.

According to Dahua Lin, Jianxiong Xiao, Xiao, this paper focused on developing a layout model for outdoor images. This paper discussed three models i.e., Descriptive, Discriminative and Generative and the reason behind for their non-consideration. Descriptive models talked about describing the scene in a whole view. For example, the Spatial Pyramid Matching talks about identification about an object and the possible outcomes. This model failed as it couldn't help in the tasks of annotation and segmentation.

Discriminative models which are mentioned here followed Conditional Random Fields (CRFs). The CRF's is a model which predicts the labels when a set of inputs (images) are fed to the system. But this fails as it doesn't have a general distinction between objects as they are talking about outdoor images. Then finally, the generative models use Hierarchical Bayesian Models which expresses relations between images. All the above-mentioned models failed because they didn't give any distinction of objects of the outdoor image, Division of images based on label's topic distribution and finding the labels beyond the images.

The experiments are conducted on three applications,

i. Scene Classification: It is based on the layout of the image
ii. Semantic Segmentation: It is based on the label’s topic distribution
iii. Layout Hallucination: To understand beyond the segmented objects of the image.

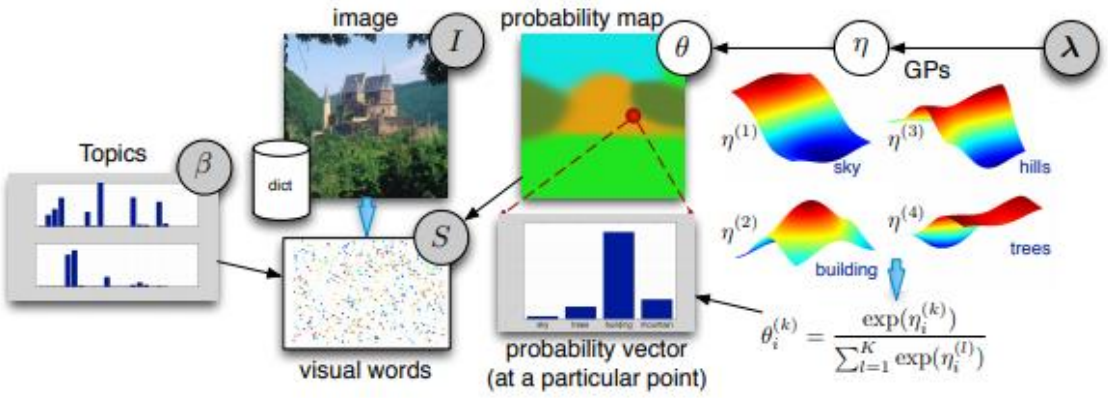

Fig 2.3 Workflow to label the objects in an image

This paper concentrates on Gaussian Processes (GP) which give layout variation and label's topic distribution. Spatial Topic Processes is a probabilistic model where it generates a continuous map of topic distributions.

$S = \{(x_i , y_i , w_i)\}$ where i value starts from 1 to n

x_i and y_i are attached to an indicator z_i which is assigned to a particular region of the image. w_i brings out the words based on the region. They explain the process in three steps.

i. Spatial Topic Process in a region
ii. Coupling Grided Gaussian Processes
iii. Joint Formulation

Initially, it should have continuous maps of topics and then have a sample of regions. Then finally, assign an indicator to the location and draw the words.

i. It is a stochastic process meaning, the model has a definite input format and output format but the process has many variations based on the Gaussian Processes.
ii. Here, the values obtained are translated into practical use. Where the process get it's mean and covariance values.
iii. This joint Formulation gives the visual words which can be derived based on the values obtained in the earlier step.

The disadvantages of this approach are, it is limited for outdoor images. When it comes to various objects, finding words gets difficult. These images with an increase in objects, they have the chance of incorrect image retrieval. Putting in mind the disadvantage, the below paper identifies objects in any annotated image regardless of the layout of the image.

According to Geremy Heitz, Daphne Koller, they use the probabilistic model to detect the “things”. This paper talks about the difference between stuff and things. Stuff here refers to materials and things to objects. To be clearer, a material is something which does not have shape but an object has a particular shape and size.
This paper has a unique advantage from the above papers that i.e., differentiating material and object. In a simplified version, the sky is considered as a material but the road is considered as an object. Therefore, the probabilistic model is called the Things and Stuff (TAS) model. This model uses Expectation-Maximization algorithm. It uses ground truth and supervised labels for training.

TAS Context: This probabilistic model should find a relationship between things and stuff. So, it has two steps. First is sliding window protocol and second is unsupervised image region clustering. A Sliding Window Protocol is a window which slides over image horizontally and in the symbol of fashion, it covers all the images. It uses conditional probability where the probability is between things and the window size. In the second step, the image is segmented to superpixels. Superpixels are a set of pixels with similar colours and the advantages or it is easier to compute features on more meaningful regions and reduce the input entities for algorithms.

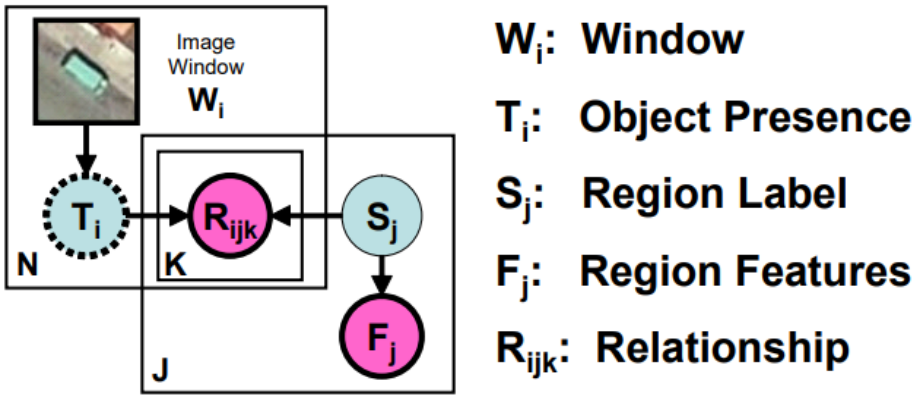

Fig 2.4 TAS Plate model

TAS model explains the object in the window and stuff in another image region and finds the relation between them. The relation is derived from the standard Bayesian model and features extracted from stuff are generated from Gaussian distribution. Combinations of various windows and various image regions from TAS ground network.

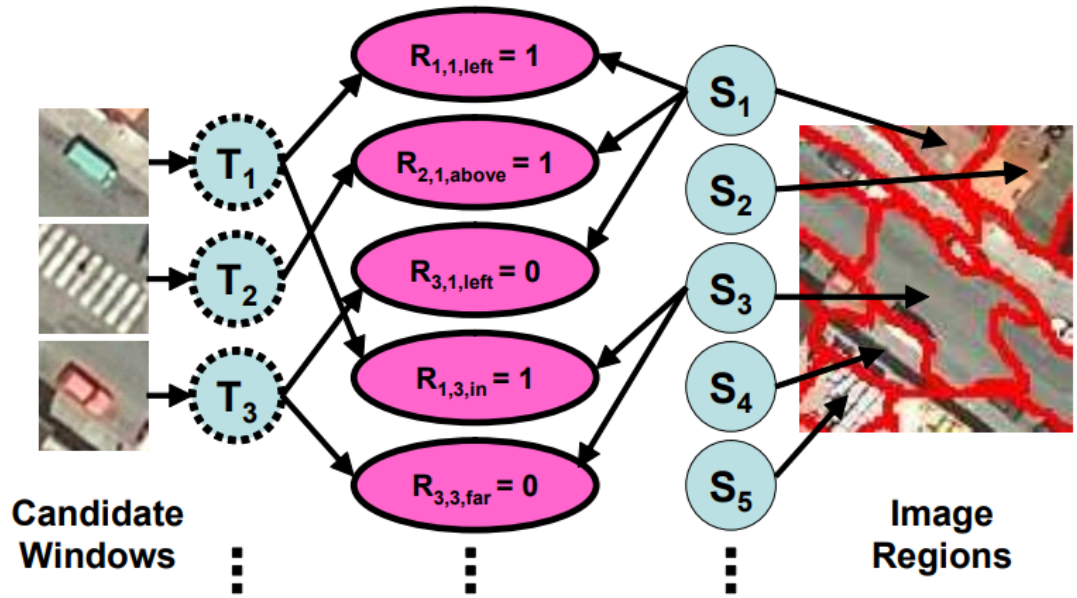

Fig 2.5 TAS Ground Network

Learning the parameters of the model is done by an E-M algorithm and further inferred by Gibbs sampling. Initially, they learn the parameters with Execution-Maximization algorithm by taking a set of candidate windows and ground truth. Every window has a set of a feature vector, relationship variable. As all the variables

are unique, they use Execution-Maximization algorithm. In the execution steps, they find the random variables of stuff (Sj) with the help of T variables. In the Maximization step, it clusters all the variables of a window to a θ_R set.

```
Algorithm Learn TAS
Input: Candidate Relationship C,Dataset D
={(W[m],T[m],F[m],R[m])}
R←Φ (all relationships"inactive")
Repeat until convergence
Repeat until convergence(EM over Parameters)
E Step:Q[m]←P(S|T,F,R;θr)
M Step:θ←argmaxEi[∑l(S,T,F,R|W;θ)
```

Fig 2.6 Algorithm for TAS

Gibbs Sampling works in the testing set. It basically finds the probability of target object in each window based on the feature of image regions and relationships.
This model has applications like a unique detection for both objects and stuff. The disadvantages of this model, it cannot detect the 3D context images.

According to Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, this paper deals with ImageNet dataset which is trained by Deep convolutional neural networks. this paper constitutes on reducing overfitting on fully connected layers by regularization method called "drop out".

To get a good percentage of accuracy in the testing set, they use datasets which contain a large number of images. Often learning datasets with a large number of images can be a problem to the system, how to overcome this they use faster GPUs. However, data sets like ImageNet data sets have large complexity of objects and the model should have the knowledge to handle all the data. This is where Convoluted Neural Networks (CNN) come to the play. CNN can be controlled by wearing the number of hidden layers and the number of input neurons. Another advantage of CNN, it needs much lesser connections and parameters to train compared to

feedforward neural networks. The CNN used here has 5 convolutional layers and 3 fully-connected layers.

ImageNet dataset has 15 million labelled images with 22000 categories. All the images are labelled using Amazon's Mechanical Trunk (AMT) tool. One of the major error rates, this data set faced in earlier experiments top-5 error rate which means it does not have a correct label among 5 labels which are most expected for the image.
Architecture: 8 learned layers - 5 convolutional layers and 3 fully-connected layers. ReLU is like an activation function which is linear for positive values and zeroes for all negative values. That is why it is called a non-linear function. The added advantages are that they take less time to compute or train as it does not have a complicated math equation. Comparing to other functions like tanh, sigmoid, it doesn't have a vanishing gradient. This network is trained in 2 GPUs as it is difficult to train in a single GPU due to the presence of large data sets. it uses parallelization scheme which essentially means that it can divide the neurons between them and they can communicate in certain layers. Pooling layers are used between convolutional layers in a CNN structure. the function of pooling layer is generally to reduce the size of the neurons which will, in turn, reduce the computation in the network. It is also used to control overfitting. This paper uses overlapping pooling if the number of neurons is different from the size of the next layer. This, in turn, will reduce the size of the neighbourhood layer for obtaining overlapping pooling. Finally, the output of the fully connected layer is fed into a 1000-way softmax. A 1000-way software produces thousand class labels.

Reduce Overfitting: They use "dropout" method to reduce overfitting. This technique is very simple where the output of each hidden neuron is set to 0 which generally has a probability of 0.5. The neurons which have a probability of 0 generally do not pass through the hidden layers and they're dropped out. this method is generally used in the first two fully connected layers so it will avoid overfitting.
Results: Using of CNN have decreased the error rate from 28.2% to 17%. As the number of convolution neural network started increasing the top 5 error rate started decreasing.

The major disadvantage is that the efficiency decreases when one of the layers are removed. The other major disadvantage they face is that they just identify about the objects identified in an image but doesn't show the stuff. To be clearer, it shows objects like a book, a chair etc but doesn’t identify walls and roads.

3. SYSTEM DESIGN

The project is divided into two parts, Visual Relationship Detection and Scene Parsing. Visual Relationship Detection is about detecting the objects and predicting the relationship among them. The visual relationship is in the form of visual phrases. Apart from spatial relationships, it also deals with non-spatial relationships such as push, taller than and so on.

Scene Parsing deals with segmenting images where they are segmented in meaningful divisions. They identify objects and their accuracies and count the number of objects. Various libraries used are MXNet, Gluon CV, OpenCV, matplotlib to avail services. The neural network used here is DeepLab ResNet 101 and VGG-16 which are pre-trained models.

3.1 ARCHITECTURE

3.1.1 DeepLab ResNet 101

DeepLab is one of the most prominent techniques for semantic image segmentation with Deep Learning. Semantic segmentation is understanding an image at the pixel level, then assigning a label to every pixel in an image so that the pixels with the same label share certain characteristics. This model was developed by Google and was open-sourced from 2016. Multiple improvements have been made to the model since then, including DeepLab V2, DeepLab V3 and the latest DeepLab V3+.

DeepLab proposes a new residual block for multi scale feature learning. Instead of regular convolutions the last block uses Atrous Convolution. The type of pooling used in this architecture is Atrous Spatial Pyramid Pooling.

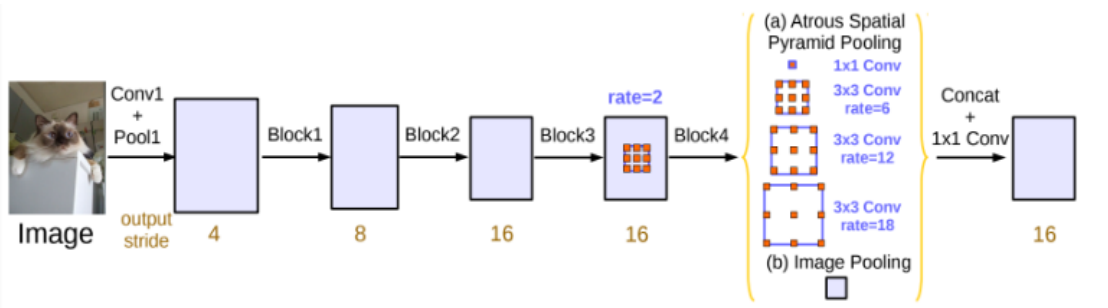

Fig 3.1 Architecture of DeepLab ResNet101

Any DeepLab model is composed of two steps:

i. Encoding phase: The aim of this phase is to extract essential information from the image which is done using Convolutional Neural Networks. The Convolutional layers present in the network look for different features for information and subsequently forward the information to the next layers. The segmentation task with this essential information classifies the image.
ii. Decoding phase: The aim of this phase to reconstruct an image with needed dimensions with the help of information from the Encoding phase.

Dilated/ Atrous Convolution*:*

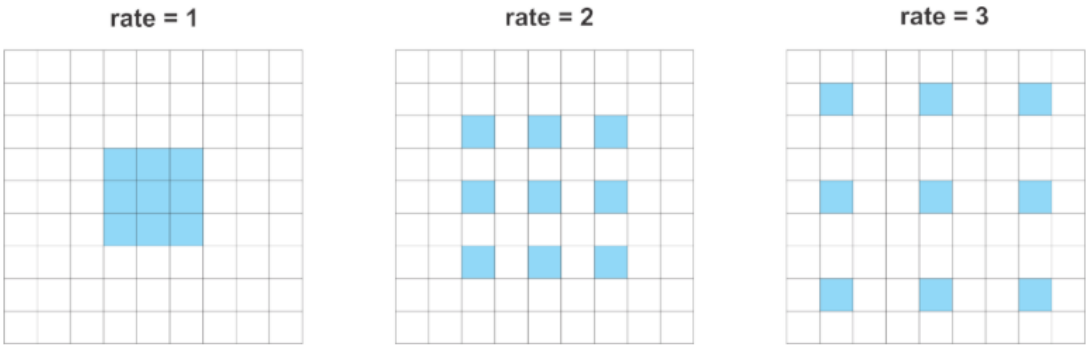

Fig 3.2 Atrous Convolution with Different Rates r

These are regular convolutions with a factor which will expand the filter's field of view. For example, if the rate=2, the filter formed is 5x5 filter. This is in turn will enlarge the convolution kernel.

As a consequence, during convolution, the filter will cover 5 cells at one stride instead of three. This will result in consuming less computational power and it is less time-consuming. The efficiency of Atrous convolution depends on dilation rate. It doesn't implement downsampling. As a result, it allows us to learn features from multi scale context using larger dilation rates. The new Atrous Residual Block contains three residual units. In total, the 3 units have three 3x3 convolutions. It has different dilation rates for each convolution.

3.1.2 Atrous Spatial Pyramid Pooling

Atrous Spatial Pyramid Pooling (ASPP) adds a series Atrous convolutions with different dilation rates. These rates are used to capture long-range context. ASPP incorporates image-level features using Global Average Pooling (GAP). Also, to add more global context information, ASPP incorporates image-level features. First, GAP is applied to the output features from the last Atrous block. Second, the resulting features are fed to previous convolution layer. Finally, the result is bilinearly up-sampled to the correct dimensions.

3.1.3 Residual Network (ResNet)

Increasing network depth does not work by stacking layers together. Deep networks are hard to train because of the gradient problem—as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitely small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.

ResNet also called as Residual Neural Network introduced its architecture with “skip connections” and also features heavy Batch Normalization. These skip connections are also known as gated units or gated recurrent units. It gives around 94 percent accuracy in top-5 and 79 percent accuracy in top-1.

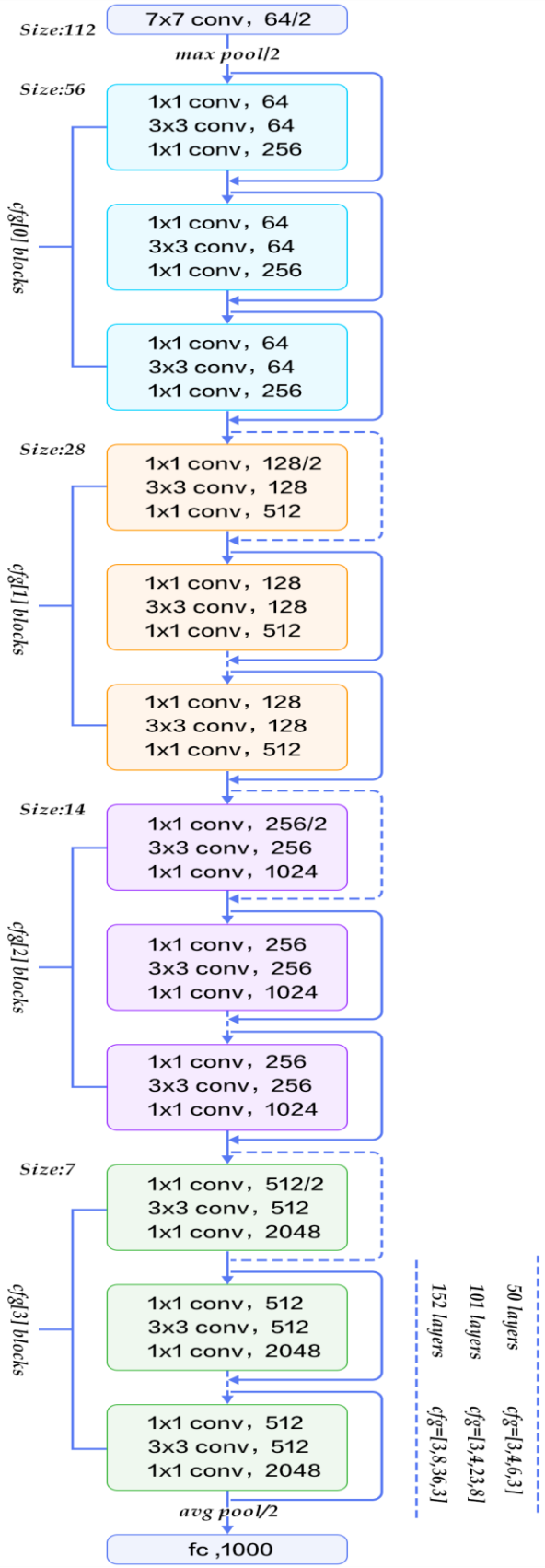

Fig 3.3 Architecture of Residual Neural Network - 101 layers

Addition of extra layers to any neural network causes degradation problem, i.e., despite adding extra layers, it lowers the training accuracy. This neural network overcomes this problem, with Residual Units.

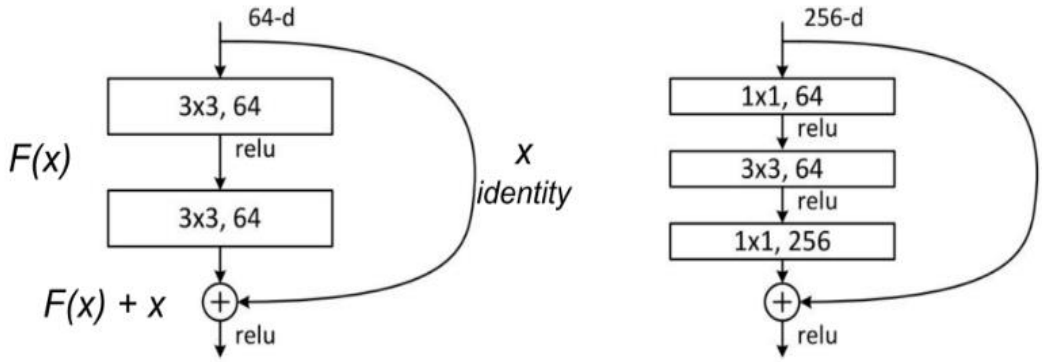

Fig 3.4 Types of Residual Units

The Neural Network contains two types of Residual units. They are

i. Baseline Blocks

ii. Bottleneck Blocks

Baseline Blocks contain two 3X3 convolutions with Batch Normalization followed by ReLU activations and Bottleneck Blocks consists of three stacked convolutions where the first layer, 1X1 is designed for reducing and restoring the dimensions. Subsequently, this leaves the next layer, 3X3 has less dense feature factor. Also, Batch Normalization is applied after each convolution and before ReLU non-linearity.

To understand the function, F(x), after the non-linear transformations in F(x), the unit combines the result of F(x) with the original input *x*. This combination is done by adding the two functions. Merging the original input *x* with the non-linear function F(x) offers some advantages. It allows earlier layers to access the gradient signal from later layers.

3.1.4 VGG Neural Network

VGG-16 is a simpler architecture model, type of convolutional neural network. It always uses 3X3 filters with stride of 1 in convolution layer and uses same padding in pooling layers with stride of 2. The input to VGG-Net is a fixed size RGB image. The image is then passed through a stack of convolutional layers where a filter size of

3X3, stride of 1 is used. Spatial pooling is carried out by five max pooling layers which are followed by convolutional layers in between. Max pooling is performed with a filter size of 2X2 using a stride of 2. The network has the general accuracy of 70.5 percent in Top-1 accuracy and 90.5 percent in Top-5 accuracy.

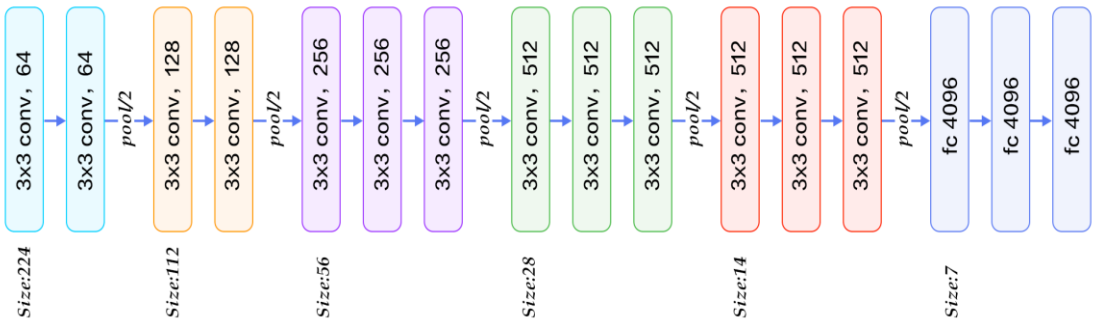

Fig 3.5 Architecture of VGG-16 Network

This network is developed and trained by Oxford's renowned Visual Geometry Group (VGG) and achieved good performance in ImageNet dataset. This network is famous for Image Localization and Image Classification task where Image Localization is finding where in the image a certain object is, described by a bounding box and classification is describing what the object in the image is.

In the first layer of this network, that is Conv1-1. It obtains 9.4% error rate, which means the additional three 1X1 convolutional layers help the classification accuracy. 1X1 convolution actually helps to increase non linearity of the decision function. Without changing the dimensions of input and output, 1X1 conv is doing the projection mapping in the same high dimensionality. Additionally, it gets an error rate of 8.4% by adding extra layers which means that the network improves when additional layers are added.

3.2 SOFTWARES USED

3.2.1 Google Colaboratory

One of the cloud services provided by Google which is similar Jupyter Notebook and this supports free GPU and Disk Space. It provides almost 12GPU space and 300GB Disk Space for free of cost. This platform allows to develop deep learning applications like TensorFlow, Keras, PyTorch, OpenCV and so on.

It supports languages used in deep learning applications like Python 2.7 and Python 3.6. Unfortunately, this application doesn't support languages like R, Scala. The working is similar to Jupyter Notebooks. It allows to create python notebooks that is .ipynb files and store notebooks, share notebooks, mount files from Google drive, upload notebooks from Git and Kaggle files. Another addition for this platform is GPU Computing. The GPU used in the backend is K80.

3.2.2 MATLAB

MATLAB, also known as Matrix Laboratory developed by MathWorks. This application is built around the MATLAB Scripting language. It is a high-performance language used for technical computing. It includes computation, visualization and programming and the problems and solutions can be easily represented in mathematical equations.

This system contains five parts where the applications runs on,

i. MATLAB Language: This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

ii. MATLAB Environment: This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and

exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

iii. Handling Graphics: This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

iv. Mathematical Functional library: This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

v. Application Program Interface: This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

3.3 SYSTEM WORKFLOW

System Workflow talks about the work proceedings of the project. It represents the design in the form of flowcharts and explains in simple steps, how the project proceeds.

3.3.1 Visual Relationship Detection

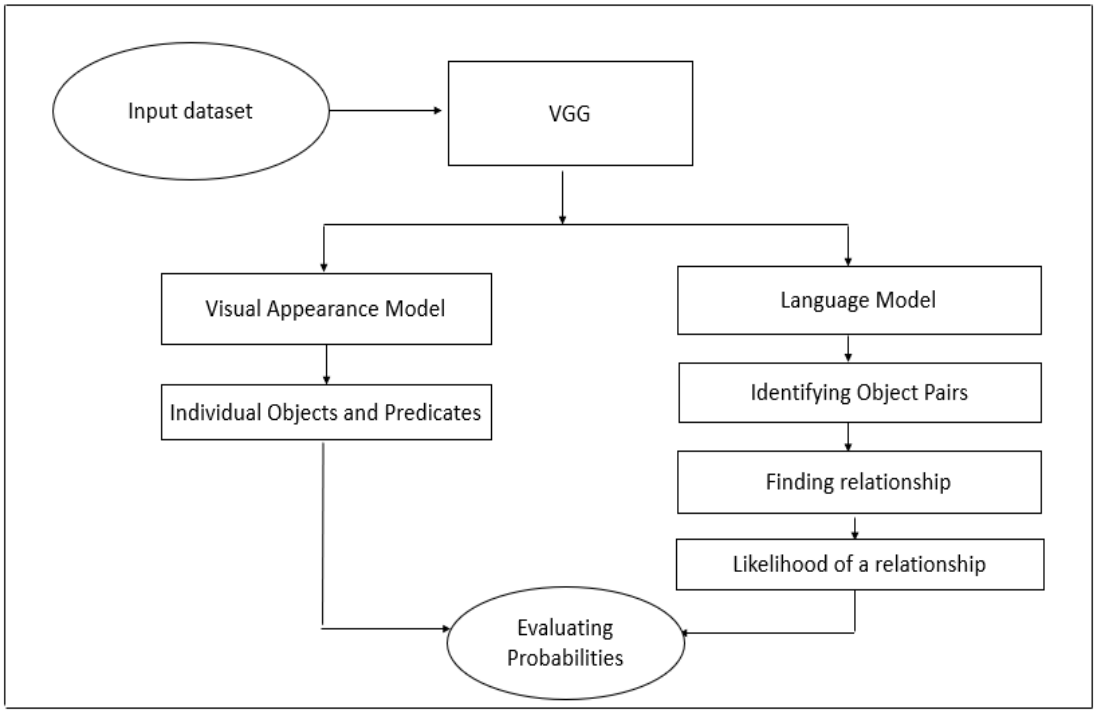

Fig. 3.6 Visual relationship Detection Design

The above flowchart talks about the Visual Relationship Detection works. Initially, the dataset used here is Scene Graph dataset which contains 5000 images with 100 object categories and 70 predicates. This dataset is trained by pre-trained neural model i.e., VGG neural network. Later, this procedure is handled in MATLAB. The model is divided into Visual Appearance Model which identifies objects and their predicates. The Language Model identifies the object pairs and with the help of word embeddings, it finds the most probable relationship. Finally, the output has a set of relationships where the relationship with highest confidence score is printed.

3.3.2 Scene Parsing

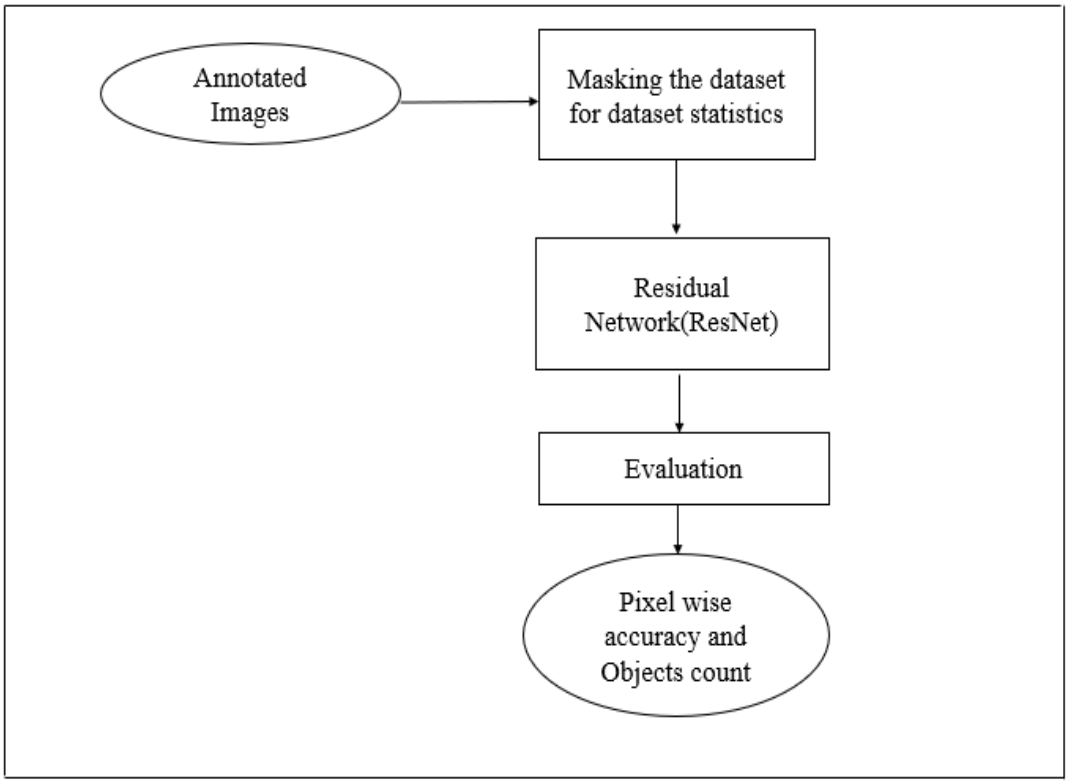

Fig 3.7 Scene Parsing Design

The above flowchart talks about Scene Parsing works. Initially, the images taken for testing are standardized according to the pre-trained Model ResNet. Then, the images are sent for testing in the pre-trained neural network model. Subsequently, the result of this model is the count of objects and segmentation of images.

3.4 REQUIREMENTS

3.4.1 Hardware Requirements

i. Quad core Intel Core i7 or higher
ii. 16GB of RAM
iii. GPU: AMD Radeon R5 M430 2 GB(Optional)

3.4.2 Software Requirements

i. Google - Colaboratory
ii. MATLAB (Tested with 2014B)

4. IMPLEMENTATION

4.1 MULTICLASS OBJECT DETECTION

Visual relationship detection involves detecting objects and finding the relationships among them. Apart from spatial relationships, it also deals with non-spatial relationships such as push, taller than. The relationship is in the form of <object1-predicate-object2>. The project deals with predicting relationships in the form of visual phrases. Different interactions can be captured between pairs of objects. These interactions can be verbs, prepositions, prepositional phrases.

The dataset used is a benchmark dataset - Scene Graph consisting of 5,000 images with 100 object categories and 70 predicates. There are in total of 37,993 relationships. 4000 images have been used for training and 1000 images for testing. There are two ways of predicting relationships. One is phrase detection and the other is relationship detection. In each method of detection, there is a Visual model and a Language model.

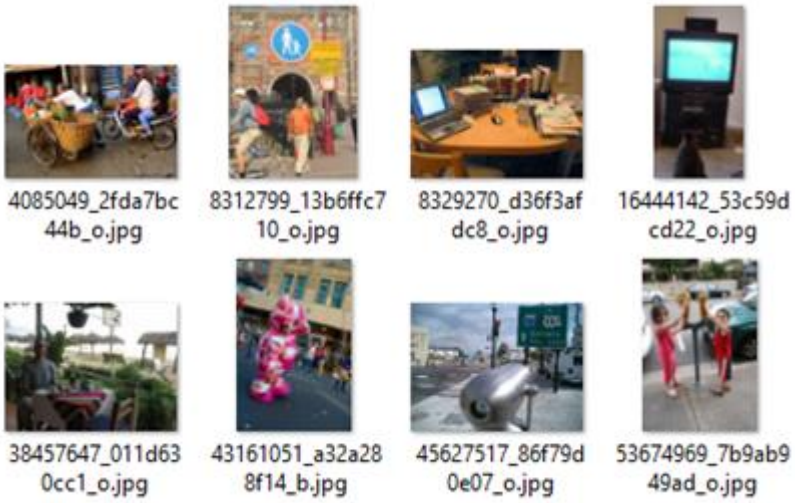

Fig 4.1 Scene Graph dataset

4.1.1 Visual Model

The input to the model is a set of images with relationship annotations where objects are localised as bounding boxes. Initially, a convolutional neural network is trained to classify 100 objects, a second neural network is trained classify 70 predicates per object category. Using the union of boundary boxes of two different object categories, the possible predicates are found.

4.1.2 Language Model

Language model is used to find relationships which are semantically close to each other. From a relationship, it should be able to infer to other relationship where both the relationships differ by only one object. The language model constitutes of three different functions projection function, training projection function and the likelihood of a relationship.

i. Projection function: In this function, a pretrained model obj2vec is used for detecting objects into two different word space. This is used for predicting the different combinations of relationships.

$$f(R_{(i,k,j)}, W)= w_k[word2vec(t_i),word2vec(t_j)]+b_k \quad \textbf{Eq. 4.1}$$

Where Eq. 4.1, a) t_i,t_j - subject and object categories.

b) b_k - bias term

c)w_k is a 600 dim and

d)word2vec is computed for a probability score.

ii. Training projection function: It is used to optimise the projection function such that similar relationships are predicted close to each other. For example, a man riding horse is close to man riding elephant but is far away from person wearing shirt. The distance between two relationships is proportional to the word2vec distance between its component objects and predicate.

iii. Likelihood of relationship: The model predicts the probability for a relationship which is more likely to occur than to a relationship which is infrequent. The visual

model and language model are combined to maximize the rank of the ground truth relationship R with bounding boxes O_1 and O_2 using the following rank loss function:

$$L(W)=\Sigma_{\{R,R'\}}\max\{f(R',W)-f(R,W)+1,0 \quad \textbf{Eq. 4.2}$$

Where Eq, 4.2, max operator provides correct ranking. R is the relationship including objects and predicates. R and R' are two relationships predicted to check which relationship should be assigned higher scores.

```
load('data/objectListN.mat');
load('data/obj2vec.mat');
load('data/UnionCNNfea.mat');
load('data/objectDetRCNN.mat');
load('data/Wb.mat');
```

Fig 4.2 Loading the data from the folder 'data'

The list of object categories are given to the model for detecting objects in a relationship. The bounding boxes are also given for all the objects. These are given as input for predicting the relationship.

```
testNum = 1000;
fprintf('#######  Relationship computing Begins  ####### \n');
for ii = 1 : testNum
  if mod(ii, 100) == 0
      fprintf([num2str(ii), 'th image is tested! \n']);
  end
  rlp_labels_ours{ii} = [];
rlp_confs_ours{ii} = [];
sub_bboxes_ours{ii} = [];
obj_bboxes_ours{ii} = [];
detL = double(detection_labels{ii});
detB = double(detection_bboxes{ii});
detC = double(detection_confs{ii});


uu = 0;
  for k1 = 1 : size(detection_bboxes{ii},1)
      for k2 = 1 : size(detection_bboxes{ii},1)
          if k1 ~= k2
          uu = uu + 1;
              % language module
              languageModual =  [W,B]*vec_org';
```

```
                % visual module
                visualModual =
detC(k1)*detC(k2)*max(UnionCNNfea{ii}(uu,:),1) ;
                rlpScore = (languageModual').*visualModual;
```

Fig 4.3 Relationship Detection

In relationship detection, input files are loaded with object categories and predicate categories. A set of images are given for detecting the relationships in a for loop. Initially, the subject and object boxes, confidence for every relationship in the image are defined. In the pretrained model, it has subject and object boundary boxes. Consider $object_1$ as k_1 and $object_2$ as k_2. For every object found, obj2vec embedding is computed for a probability score. Using these two objects, all the possible set of pairs of objects are found. Using the given set of subjects, objects and predicates which were found earlier, possible relationships are predicted. The relationships may or may not be true about the image. Relationship score is computed for every relationship. The relationship scores are stored as confidence for every relationship. The model assigns highest score for the relationship which is true generally.

```
[m_score, m_preidcate]  = max(rlpScore);
rlp_labels_ours{ii} = [rlp_labels_ours{ii}; [detL(k1),
m_preidcate, detL(k2)]];

rlp_confs_ours{ii} = [rlp_confs_ours{ii}; m_score];
sub_bboxes_ours{ii} = [sub_bboxes_ours{ii};detB(k1,:) ];
obj_bboxes_ours{ii} = [obj_bboxes_ours{ii};detB(k2,:) ];
```

Fig 4.4 Finding highest relationship score

At present, every pair of objects i.e., object1 and object2, all predicates are computed for the most probable relationships. Based on UnionCNN feature, the predicate which has the highest score is chosen. The relationship assigned to this score will be displayed for the output image.

```
For ii = 1 : length(rlp_confs_ours)
    [Confs, ind] = sort(rlp_confs_ours{ii}, 'descend');
    rlp_confs_ours{ii} = Confs;
    rlp_labels_ours{ii} = rlp_labels_ours{ii}(ind,:);
    sub_bboxes_ours{ii} = sub_bboxes_ours{ii}(ind,:);
    obj_bboxes_ours{ii} = obj_bboxes_ours{ii}(ind,:);
end
save('results/relationship_det_result.mat', 'rlp_labels_ours',
'rlp_confs_ours', 'sub_bboxes_ours', 'obj_bboxes_ours');

fprintf('\n');
fprintf('#######  Top recall results  ####### \n');
recall100P = top_recall_Phrase(100, rlp_confs_ours,
rlp_labels_ours,
sub_bboxes_ours, obj_bboxes_ours);
recall50P = top_recall_Phrase(50, rlp_confs_ours,
rlp_labels_ours,
sub_bboxes_ours, obj_bboxes_ours);
fprintf('Phrase Det. R@100: %0.2f \n', 100*recall100P);
fprintf('Phrase Det. R@50: %0.2f \n', 100*recall50P);
```

Fig 4.5 Storing the confidence for every relationship in results folder

For example, if the model has generated 13 relationships. Every relationship generated will have a predicate. The confidence of every relationship is same as that of relationship score. The predicate with highest score is found. For all the relationships detected, the scores are stored in the decreasing order and the scores are stored in a file. For every relationship in an image, the subjects and objects are identified with the help of boundary boxes.

```
ob1.cx = round((box1(2) + box1(4))/2);
ob1.cy = round((box1(1) + box1(3))/2);
ob2.cx = round((box2(2) + box2(4))/2);
ob2.cy = round((box2(1) + box2(3))/2);
rel.cx = round((ob1.cx + ob2.cx)/2);
rel.cy = round((ob1.cy + ob2.cy)/2);
gcf=figure;
imshow(im);
text(ob1.cy,ob1.cx,strObject1,'color','red','fontsize',20);
text(ob2.cy, ob2.cx,strObject2,'color','blue','fontsize',20);
if size(im,1)>size(im,2)
  strRep=['<',strObject1,',',strRelationship,',',strObject2
    ,'>score:', sprintf('%0.1f',rlp_confs_ours{id}(idx))];
text(1,round(size(im,2)/10),strRep,'color','green','fontsize',
18)
```

Fig 4.6 Displaying the output image

This is used to display the relationship output in a boundary box in the image. The boundary boxes are calculated for subject and object. The values are given in the form of X-axis maximum, minimum value and Y-axis maximum and minimum value. The subject is represented using red colour, the object using blue colour and the relationship using green colour along with the relationship score.

Only that relationship with highest score will be shown. The output image is saved as .png file.

```
tp_cell{i} = tp;
fp_cell{i} = fp;

t = tic;
tp_all = [];
fp_all = [];
confs = [];
for ii = 1 : num_imgs
tp_all = [tp_all; tp_cell{ii}(:) ];
fp_all = [fp_all; fp_cell{ii}(:) ];
confs = [confs; tuple_confs_cell{ii}(:)];
end

[confs, ind] = sort(confs,'descend');
tp_all = tp_all(ind);
fp_all = fp_all(ind);
tp = cumsum(tp_all );
fp = cumsum(fp_all );
recall =(tp/num_pos_tuple);
top_recall = recall(end);
```

Fig 4.7 To find recall of images

Recall@100 is also found which computes the fraction of times the correct relationship is predicted in the top 100 confident relationship predictions. To find recall, a zero array for true positive and a false positive are created to the size of the number of relationships for the image. The cumulative sum of true positives and false positives are stored. The top recall computes the fraction of times the correct relationship is predicted.

4.2 SCENE PARSING

Scene Parsing deals with segmenting images where they are segmented in meaningful divisions. They identify objects and their accuracies and count the number of objects. Various libraries used MXNet, Gluon CV, OpenCV, matplotlib to avail services. The neural network used here is DeepLab ResNet 101 which is a pre-trained model.

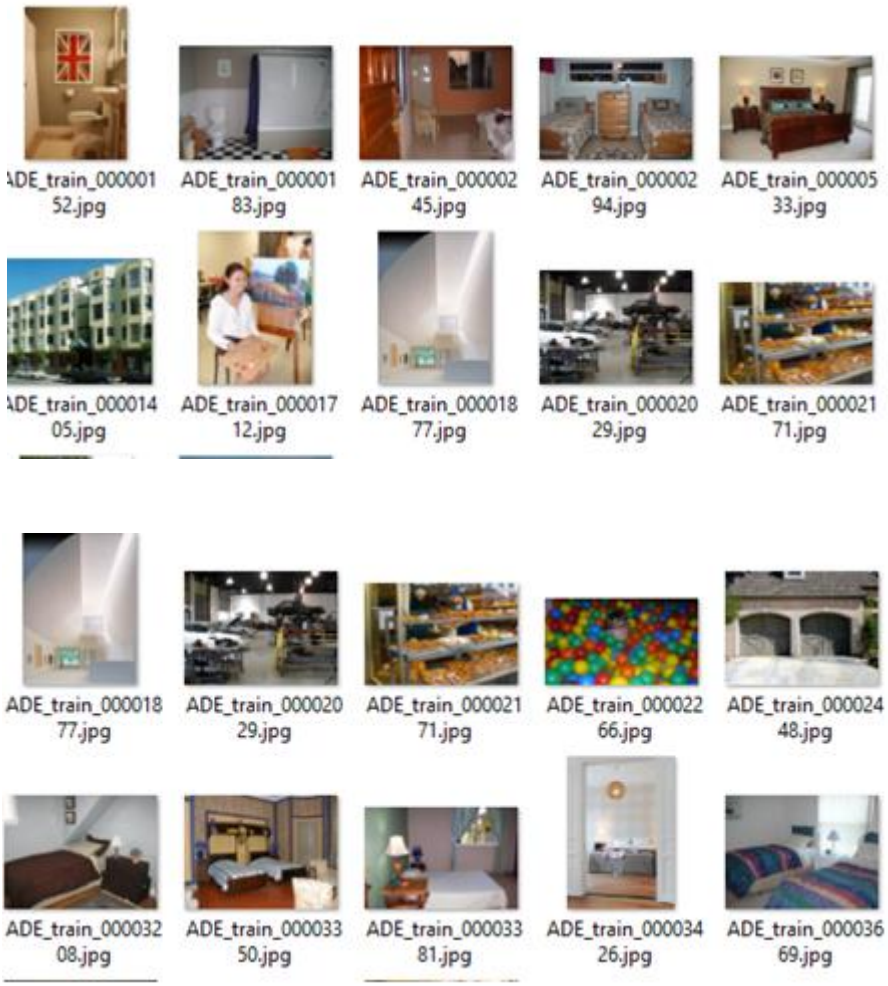

Fig 4.8 ADE20K Dataset

```
import sys
import os
import mxnet as mx
from mxnet import image
from mxnet.gluon.data.vision import transforms
import gluoncv
from matplotlib import pyplot as plt
from gluoncv.utils.viz import get_color_pallete
import matplotlib.image as mpimg
import numpy as np
import mxnet as mx
from gluoncv import data, utils
```

Fig 4.9 Libraries

Open CV Python

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. It is used for facial recognition or building a complete deep learning pipeline for image classification.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

MXNet

Apache MXNet is an open-source deep learning software framework, used to train, and deploy deep neural networks. MXNet is used to define, train and deploy deep neural networks. It is lean, flexible and ultra-scalable i.e. it allows fast model-training and supports a flexible programming model and multiple languages.

MXNet supports multiple languages like C++, Python, R, Julia, Perl etc. This eliminates the need for learning a new language to use the framework and simplify network Tdefinitions. MXNet models are portable in a manner that they are able to fit in very small amounts of memory. Hence one can train his/her model in the cloud and deploy it on either mobile or connected device. MXNet can also scale to multiple

GPUs and multiple machines. Due to its various advantages over other deep learning frameworks, MXNet has been chosen by Amazon for its Web Services' Deep Learning frameworks.

GLUON CV

GluonCV provides implementations of state-of-the-art (SOTA) deep learning algorithms in computer vision. It aims to help engineers, researchers, and students quickly prototype products, validate new ideas and learn computer vision.

GluonCV features

i. training scripts that reproduce SOTA results reported in latest papers,
ii. a large set of pre-trained models,
iii. carefully designed APIs and easy to understand implementations,
iv. community support.

Next, A function is created for image segmentation using ADE20K Dataset. For the input of images, all the images are sent to transform by normalization by particular values. All pre-trained models expect input images normalized in the same way, i.e. mini-batches of 3-channel RGB images of shape (N x 3 x H x W), where N is the batch size, and H and W are expected to be at least 224. The images have to be loaded in to a range of [0, 1] and then normalized using and std = [0.229, 0.224, 0.225]. The transformation should preferably happen at preprocessing.

```
print("Normalizing image:\t")

transform_fn =
transforms.Compose([transforms.ToTensor(),transforms.Nor
malize([.485, .456, .406], [.229, .224, .225])])

img = transform_fn(img)
img = img.expand_dims(0).as_in_context(ctx)
print(img)
```

Fig 4.10 Transformation of image

Gluon CV is a toolkit which stores all deep-learning algorithms and the pretrained models are obtained here. The Gluon Model Zoo API, defined in the gluon.model_zoo package, provides pre-defined and pre-trained models to help bootstrap machine learning applications.

```
print("Loading Model:.........\t")

model =
gluoncv.model_zoo.get_model('deeplab_resnet101_ade',pretraine
d=True)

print("Model details:\t")
print(model)
```

Fig 4.11 Pretrained model from Gluon CV

Finally, the function ends with segmenting the image with “argmax” function where the function gives maximum value to the neighbouring pixels for an object in the image.

```
print("Predicting Image using Model:\t")
print(".........................:")
output = model.demo(img)
predict = mx.nd.squeeze(mx.nd.argmax(output, 1)).asnumpy()
mask = get_color_pallete(predict, 'ade20k')
```

Fig 4.12 Segmentation of the images

The same is followed for another benchmark dataset, VOC Dataset.

The object identification is processed using a COCO Dataset pretrained model. This pre-trained model contains 80 object categories and further counting of objects is from this pretrained model. The below function talks about identifying objects and boundary boxes. Plot_bbox is a function which identifies boundary boxes with different parameters like score which is the pixel accuracy, the threshold to be shown and so on.

```
if labels is not None and not len(bboxes) == len(labels):
    raise ValueError('The length of labels and bboxes mismatch,
{} vs {}'
                     .format(len(labels), len(bboxes)))
if scores is not None and not len(bboxes) == len(scores):
    raise ValueError('The length of scores and bboxes mismatch,
{} vs {}'
                     .format(len(scores), len(bboxes)))

ax = plot_image(img, ax=ax, reverse_rgb=reverse_rgb)

if len(bboxes) < 1:
    return ax

if isinstance(bboxes, mx.nd.NDArray):
    bboxes = bboxes.asnumpy()
if isinstance(labels, mx.nd.NDArray):
    labels = labels.asnumpy()
if isinstance(scores, mx.nd.NDArray):
    scores = scores.asnumpy()
```

Fig 4.13 Boundary boxes function

Using a pre-trained model, a set of objects are already trained in this neural network and they can be found by,

```
obj_Detection =
gluoncv.model_zoo.get_model('faster_rcnn_resnet101_v1d_coco',
pretrained=True)
```

Fig 4.14 The objects in the pretrained model

Finally, the segments and objects are differentiated and the count of all objects is done through the below function. In this function, initially, the list of class labels for the objects is detected. Then the count of all the class labels is done by using the length function.

```
for i in range(len(numItems_ade)):
    print(numItems_ade[i])
    print(numItems_voc[i])
    print("Numbers Objects idenfied in an image {}:\t{}".
        format(names_of_imgs[i],len(class_names_of_imgs[i])))
  class_names_of_img_unique=list(set(class_names_of_imgs[i]))
 for j in range(0,len(class_names_of_img_unique)):
 print("Number of {}'s -------> :{}".
format(class_names_of_img_unique[j],class_names_of_imgs[i].
count(class_names_of_img_unique[j])))
```

Fig 4.15 Final Comparison

The above function explains about the count of objects. Initially, the semantic labels obtained from object identification where pretrained model trained with COCO Dataset is used. So, the count is obtained from semantic labels from boundary boxes and calculated.

5. TESTING AND RESULTS

5.1 MULTICLASS OBJECT DETECTION

Multiclass Object Detection are a pair of localized objects connected via a predicate. Visual relationships are detected by learning visual appearance models for objects and predicates and using the relationship embedding space. Understanding relationships improves image-based retrieval.

```
>> run relationship_phase_detection.m
#######  Relationship computing Begins  #######
100th image is tested!
200th image is tested!
300th image is tested!
400th image is tested!
500th image is tested!
600th image is tested!
700th image is tested!
800th image is tested!
900th image is tested!
1000th image is tested!

#######  Top recall results  #######
Phrase Det. R@100: 17.33
Phrase Det. R@50: 16.46
Relationship Det. R@100: 14.96
Relationship Det. R@50: 14.11
```

Fig 5.1 Testing and evaluation metrics

It displays ‘output image is tested’ after every 100 images. It also gives recall@100 and recall@50 which tells about the fraction of times the correct relationship is predicted out of 100 confident relationships.

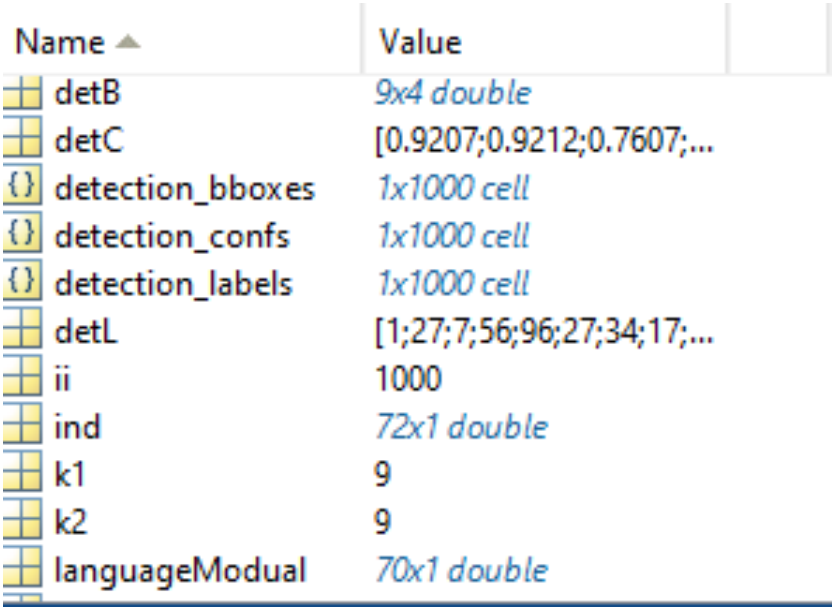

Fig 5.2 Variables shown

The output describes about the subject and object boundary boxes. It also shows about the number of infrequent relationships possible in the image. k_1, k_2 are the subject and object indices.

```
>> annotation_test{1}.relationship{1}

ans =

  struct with fields:

    subBox: [192 885 275 568]
    objBox: [292 529 342 569]
    phrase: {'person'  'wear'  'shirt'}
```

Fig 5.3 Relationship of an image

For example, an image has 13 relationships which later can be retrieved along with the subject and object box.

Fig 5.4 Visualization Results - 1

In this image, an output relationship of <car behind person> is given. Other relationships like <person wear shirt>, <car infront of person> are not displayed as they have less relationship scores.

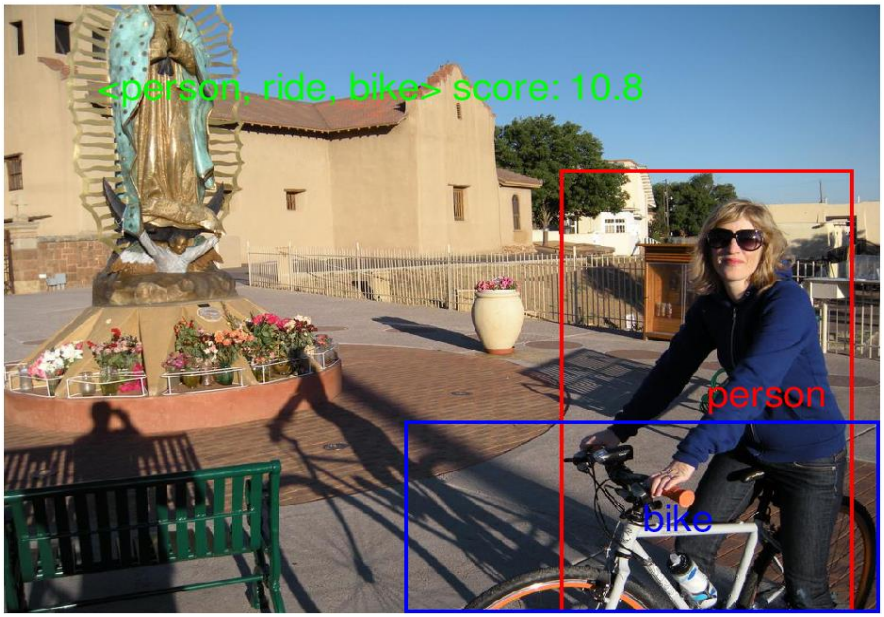

Fig 5.5 Visualization Results - 2

In this image, an output relationship of <person ride bike> is given. Other relationships like person wear shirt are not displayed as they have less relationship scores. All the suitable predicates for the set of objects in the image are found. The other relationships which are not been predicted are the infrequent relationships.

5.2 SCENE PARSING

Initially, all the input pixels of the image are shown when the function for segmentation is executed.

```
Input Image Pixels:

[[[154 137 130]
  [114  97  90]
  [ 52  35  28]
  ...
  [185 185 183]
  [185 185 183]
  [185 185 183]]

 [[142 125 118]
  [153 136 128]
  [148 131 124]
  ...
  [186 186 184]
  [185 185 183]
  [185 185 183]]

 [[148 131 123]
  [149 132 122]
  [162 145 137]
  ...
  [186 186 184]
  [186 186 184]
  [185 185 183]]
```

Fig 5.6 Input Pixels of the input image

Then the image is normalizing through standard values,

```
Normalizing image:

[[[[ 0.5193082  -0.16568205 -1.2274169  ...  1.0501755   1.0501755
     1.0501755 ]
   [ 0.31381115  0.50218344  0.41655967 ...  1.0673003   1.0501755
     1.0501755 ]
   [ 0.41655967  0.43368444  0.6563062  ...  1.0673003   1.0673003
     1.0501755 ]
   ...
   [-0.2170563  -0.37117907 -0.49105233 ...  0.7590547   0.7761795
     0.74193   ]
   [-0.7307989  -0.6109256  -0.5424266  ...  0.7761795   0.72480524
     0.673431  ]
   [-1.073294   -0.7136741  -0.6109256  ...  0.7761795   0.70768046
```

Fig 5.7 Normalizing Image

Segmentation results for ADE20K dataset and VOC Dataset,

Fig 5.8 Input Image 1

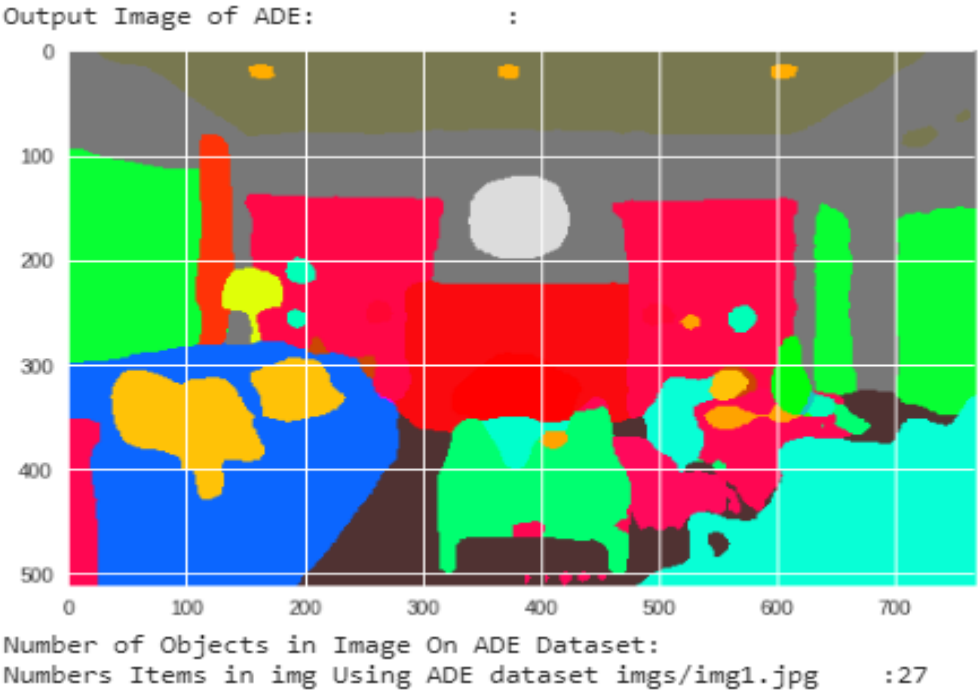

Fig 5.9 Segmentation count and Segmentation with ADE20K Dataset- 1

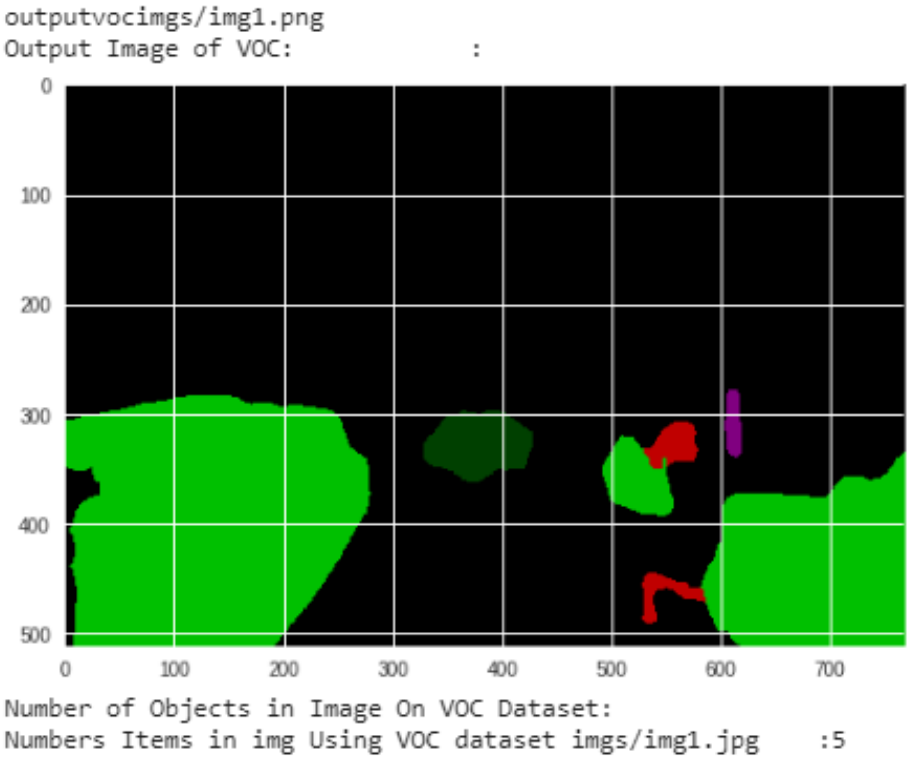

Fig 5.10 Segmentation count and Segmentation with VOC Dataset- 1

The above three images show difference between the neural networks trained by two benchmark datasets i.e., ADE20K Dataset and VOC Dataset. It is done by using two separate pretrained models where same neural network is used but it is trained with two different datasets. As the Fig 5.9 shows more segmentations than Fig 5.10 because the dataset ADE20K Dataset is trained with more in number of object categories than VOC Dataset. The output images depending on the background have various segments. Both the datasets have incorrect results for backgrounds which contain forests, animals and so on.

Fig 5.11 Input Image - 2

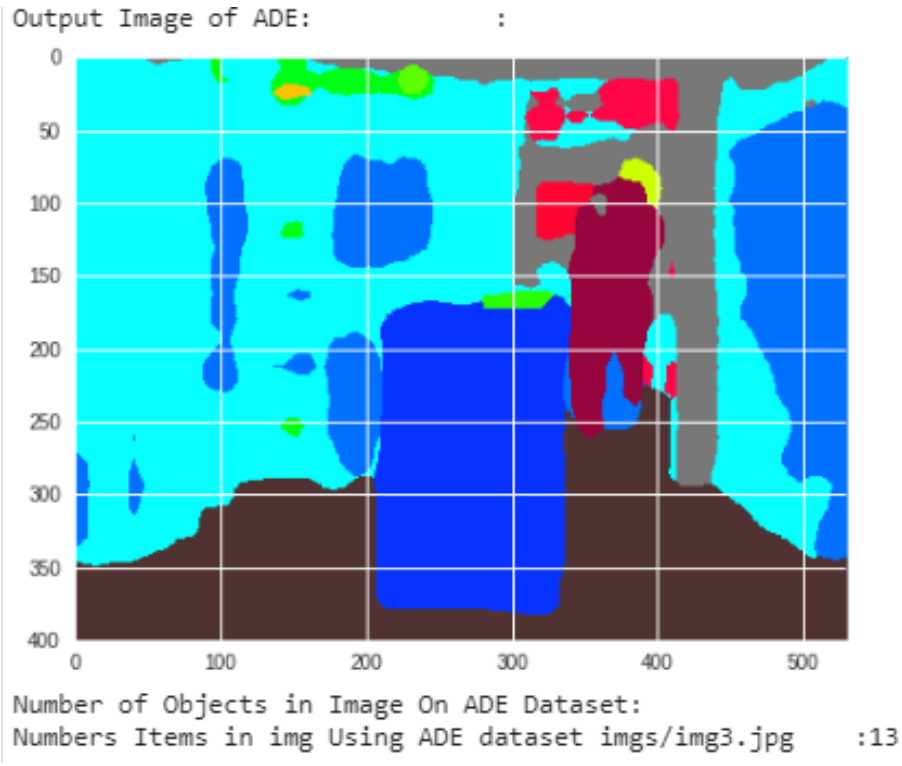

Fig 5.12 Segmentation count and Segmentation with ADE20K Dataset- 2

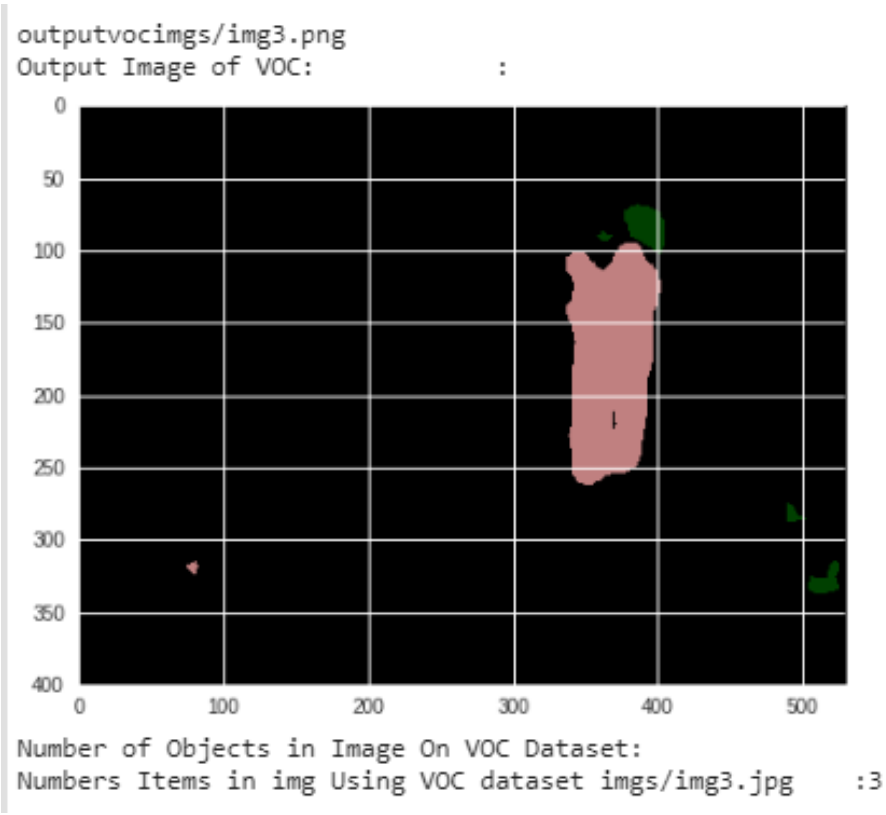

Fig 5.13 Segmentation count and Segmentation with VOC Dataset- 2

From the above, Fig 5.11 is the input image where in the results, the number of segments are more in number in Fig 5.12 than Fig 5.13 because the ADE20K dataset is trained with more object categories than PASCAL VOC dataset images.

The object identification is through COCO Dataset pre-trained model which has 80 object categories. The count of objects is actually completed using the semantic labels present when the objects are identified. But as the object categories are relatively low than ADE20K Dataset, there are some incorrect object identifications.

The below pictures show the object identifications and the subsequent counting of objects in image is due to these semantic labels of boundary boxes.

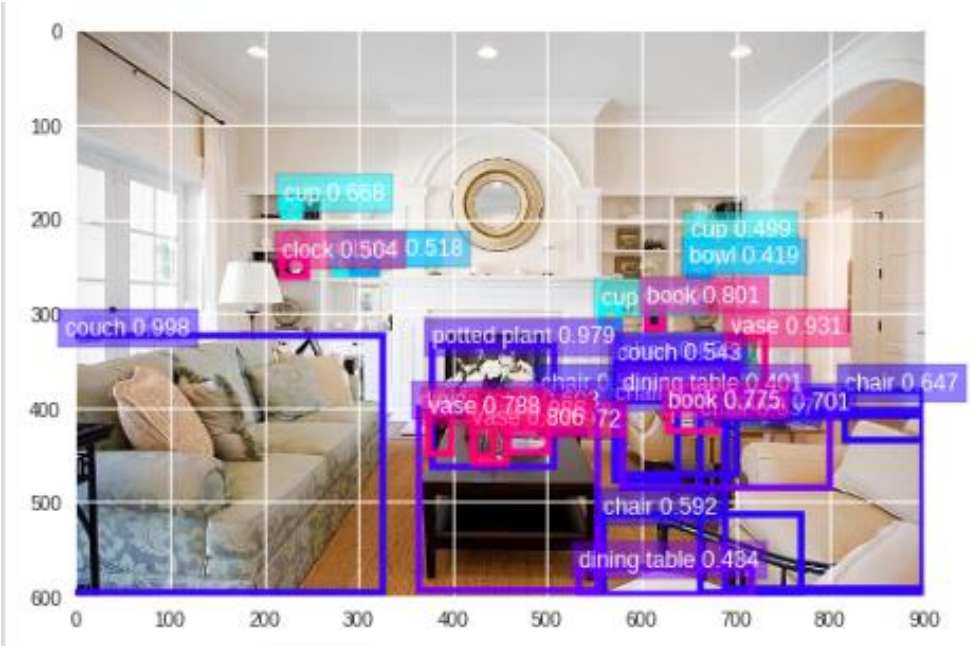

Fig 5.14 Object Identification and their accuracies – 1

The above figure, Fig 5.14, the algorithm failed to recognise various objects like door, cupboard, windows and mirror.

Fig 5.15 Object Identification and their accuracies – 2

For example, in the Fig. 5.15, the object “desk” is misidentified as “oven”. Most of the time, these misidentifications can be controlled using higher threshold values.

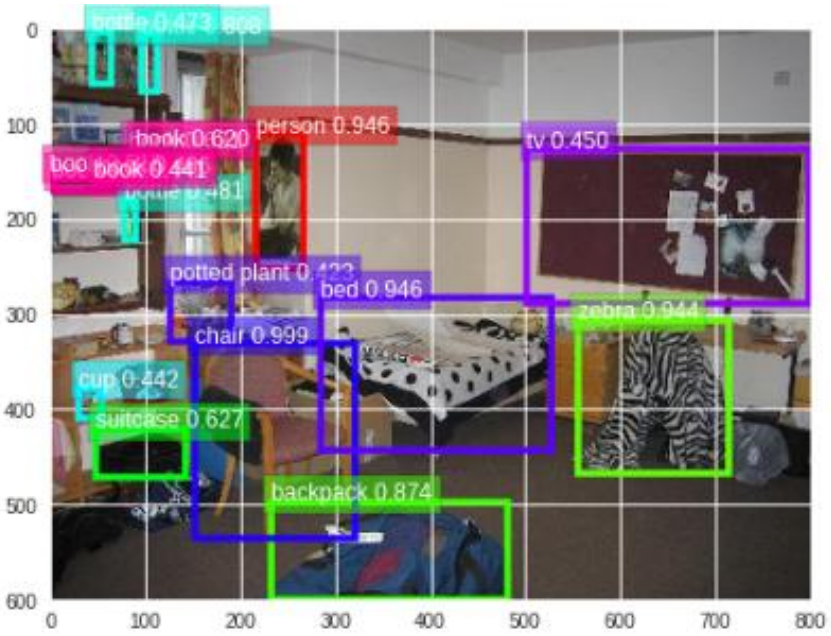

Fig 5.16 Object Identification and their accuracies – 3

The above image, Fig 5.16, the model misidentifies the object ‘bag’ as zebra due to its unique colour combination similar to the animal Zebra.

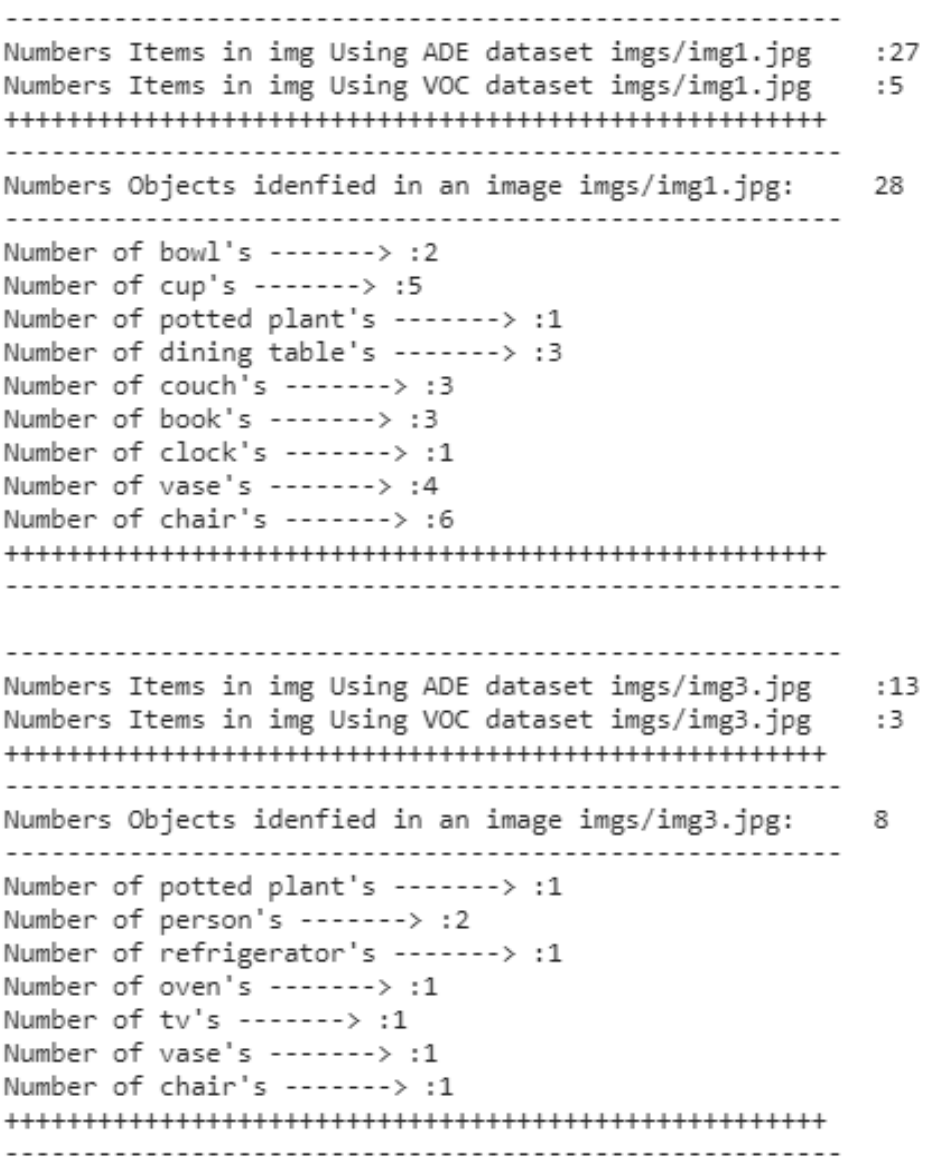

Fig 5.17 Count of objects

The count of objects is calculated from number of semantic labels produced at the object identification. As the count of images is obtained by the same algorithm used for object identification, the count of objects face the similar flaws. From, the Fig 5.14, several objects like door, cupboard, windows are missing and the count of objects is miscounted for this reason and from the Fig 5.15, the object cupboard is misidentified as oven and subsequently, the count for the oven is one.

6. CONCLUSION AND FUTURE SCOPE

Segmentation is essential for image analysis tasks. Semantic segmentation describes the process of associating each object of an image with a class label. It can be seen in many future applications like autonomous driving, Industrial inspection, Medical diagnosis and so on. Multi Object detection can identify the probable relationship between two objects by word embeddings. Word embeddings has been the core part of this model but word embeddings has its own disadvantages.

One of the major limitations of this project is the usage of the pretrained model. The usage of pretrained model will prevent the training of the image according to one's use. As the models used for object identification is also pretrained model, it has limited number of object categories. Additionally, the pretrained models trained for datasets like ADE20K datasets segment the image to objects which are not present and the images with outdoor background fail to identify the objects.

The project can be further developed by using the models which can be trained, if provided with vast amounts of GPUs. Additionally, this project can be developed in the perspective of identifying objects and its parts like the rim of a wheel. Another approach of developing would be in the matter of decreasing the count of incorrectly identified objects and achieve higher rates of accuracy.

BIBLIOGRAPHY

1. Ruslan Salakhutdinov, Antonio Torralba, Josh Tenenbaum, “Learning to Share Visual Appearance for Multiclass Object Detection” , IEEE, CVPR 2011, 8 pages.
2. Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning, “Generating Semantically Precise Scene Graphs from Textual Descriptions for Improved Image Retrieval” , VL@EMNLP 2015, 11 pages.
3. Dahua Lin, Jianxiong Xiao, “Characterizing Layouts of Outdoor Scenes Using Spatial Topic Processes” , Elsevier Expert Systems with Applications 2016, 8 pages.
4. Geremy Heitz, Daphne Koller, “Learning Spatial Context: Using Stuff to Find Things” , Computer Vision – ECCV 2008, 14 pages.
5. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks” , ILSVRC-2012, Advances in neural information processing systems, 9 pages
6. Bolei Zhou , Hang Zhao , Xavier Puig , Sanja Fidler , Adela Barriuso , Antonio Torralba, “Scene Parsing through ADE20K Dataset”, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 14 pages.
7. Cewu Lu, Ranjay Krishna, Michael Bernstein, Li Fei-Fei, “Visual Relationship Detection with Language Priors”, ECCV 2016, 19 pages.
8. V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet, “A deep convolutional encoder-decoder architecture for image segmentation.” arXiv:1511.00561, 2015, 11 pages.
9. M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. “The pascal visual object classes (voc) challenge”, Int’l Journal of Computer Vision, 2010, 36 pages.
10. J. Tighe and S. Lazebnik.” Finding things: Image parsing with regions and per-exemplar detectors.” In Proc. CVPR, 2013, 8 pages.
11. Li-Jia Li, Richard Socher, Li Fei-Fei. “Towards Total Scene Understanding: Classification, Annotation and Segmentation in an Automatic Framework.” In Proc. of CVPR’09, 2009, 8 pages.

12. Antonia Torralba, “A.: Contextual priming for object detection.” IJCV 53, 23 pages.
13. Bryan A. Plummer, Arun Mallya, Christopher M. Cervantes, Julia Hockenmaier, Svetlana Lazebnik, “Phrase Localization and Visual Relationship Detection With Comprehensive Image-Language Cues” ICCV 2017, 10 pages.
14. Ruichi Yu, Ang Li, Vlad I. Morariu, Larry S. Davis, “Visual Relationship Detection With Internal and External Linguistic Knowledge Distillation” , ICCV 2017, 9 pages.