**ImageNet Object Detection Challenge**

**Machine Learning Project Final Report**

# BY

**HARSH BHALANI(hpb170030)**

**MAITRI SHAH(mxs172030)**

**NIDHI VAISHNAV(ntv170030)**

**SHIVANI THAKKAR(sdt170030)**

# 



**Eric Jonsson School of Engineering and Computer Science**

**UNIVERSITY OF TEXAS AT DALLAS**

**FALL 2017**

**Chapter 1**

**Introduction and Problem Description**

ImageNet Object Detection Challenge is an official Kaggle Competition problem. It is an estimate that by the end of 2017, there will be 1.2 trillion images available online. If someone takes around 1 second to annotate, then also it will take around 38000 years to classify them all. Researchers have been working effectively to design efficient algorithms for automatically classification of images. But due to the limitation of good quality of datasets, many researchers are not successful to accomplish this task.

ImageNet dataset is provided which contains large number of human annotated images. This massive dataset gives all researchers an opportunity to analyze different techniques for automatic image classification. Moreover, ImageNet dataset contains variety of images containing a great amount of distinct objects.

A good commercial use of image classification can be found in the field of stock photography and video. Stock websites gives platforms to sell photos and videos. Contributors require a method to tag many visual materials, which consumes large amount of time and is even tedious.

If any visual database contains metadata about the images, a huge tedious task of categorizing them can be avoided. Classification of images with the help of machine learning is a prime solution for this. With image classification, companies can easily classify and categorize their database. This helps them to manage their visual database without investing large amount of hours for manual sorting and tagging.

**Chapter 2**

**Related work**

There are multiple evidences of work that have examined the impact of factors like occlusion, change in aspect ratios and variation in viewpoints for general as well as specific object detection as well as classification.

Deep CNNs are generally used for detection of objects of general classes. Researchers have also worked on multi-stage pipeline known by Regions with Convolutional Neural Networks i.e. R-CNN for classification of regional portions of an entire image.

It decomposes the problem of detection into various stages which includes proposal of bounding box, Convolutional Neural Network pre-training, fine tuning of Convolutional Neural Network and regression of bounding box.

GoogleNet was proposed by Google which has a 22-layer structure and modules of inception which replaces CNN by R-CNN.

**Chapter 3**

**Dataset description**

**3.1 Training data**

The training set contains around 475,000 objects for classification from around 450,000 images. Training dataset contains three folders.

* Annotations
* The image annotations are provided in XML formatted files in PASCAL VOC format.
* This folder contains different folders for years 2013 and 2014. In these folders, there are xml formatted files for each particular image.
* This xml file contains general information like image filename, folder name and size of image.
* It also contains specific information for each object contained in that image. Corresponding to each image object, its name, xmin (minumum x co-ordinate), xmax (maximum x co-ordinate), ymin (minimum y co-ordinate) and ymax (maximum y co-ordinate)
* Data
* This folder contains several folders for years 2013 and 2014.
* Each of these folders contain images.
* ImageSets
* This folder contains 200 text files corresponding to 200 categories.
* This test file constitutes paths of images belonging to that category.

**3.2 Testing data**

Testing dataset contains two folders:

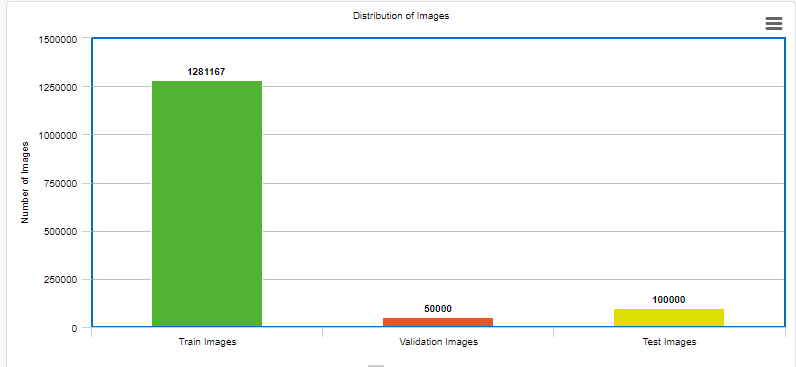
* Data
* This folder contains 65,500 images to perform testing.
* ImageSets
* This folder contains text file containing names of 65,500 images spanning across 200 categories.

**3.3 Dataset Download Links**

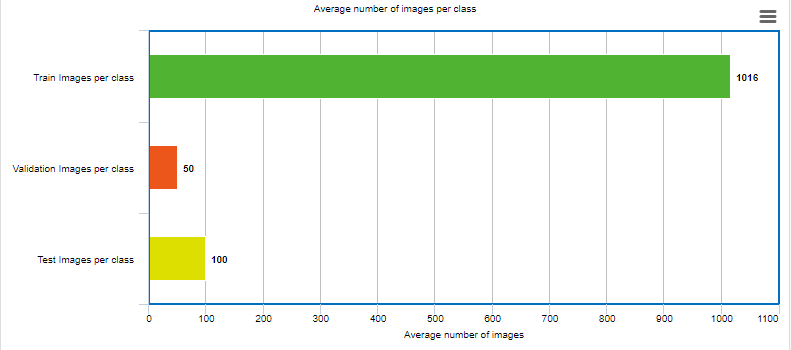
Table 3.1 Download Links

|  |  |
| --- | --- |
| Name of the File | Download Link |
| Training Data | <https://www.kaggle.com/c/imagenet-object-detection-challenge/download/imagenet_object_detection_train.tar.gz> |
| Testing Data | [https://www.kaggle.com/c/imagenet-object-detection-challenge/ download/imagenet\_object\_detection\_test.tar.gz](https://www.kaggle.com/c/imagenet-object-detection-challenge/%20download/imagenet_object_detection_test.tar.gz) |

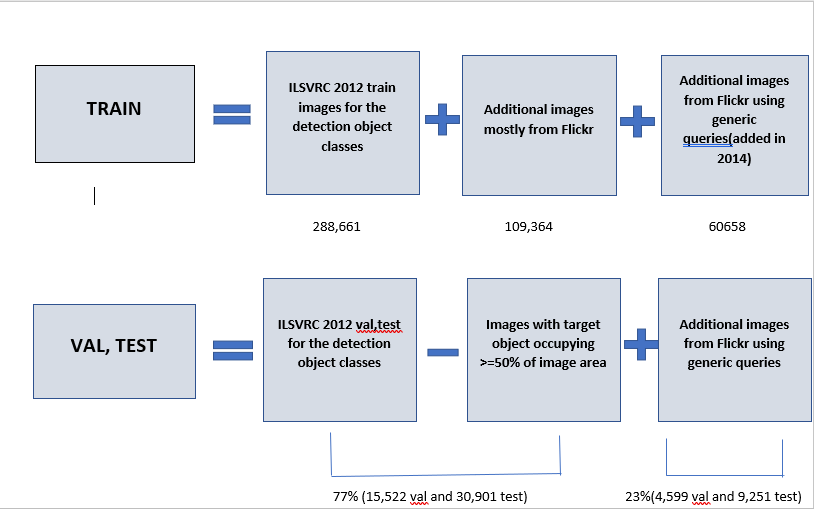
**3.4 Data Distribution**

****

**Fig. 1 Distribution of images across training, validation and testing dataset**



**Fig. 2 Average number of images per class across training, validation and testing datasets**

****

**Fig. 3 Summary of images collected for the Object Detection Task**

**3.4 Features and Instances**

In this dataset, the features used for the task of image classification is the image itself. Instances are also images.

**Chapter 4**

**Pre-processing techniques**

**4.1 Initial Idea about preprocessing**

Images in the training dataset constitutes multiple images. So, to classify different images, we first planned to extract individual objects from each image. For extraction of individual objects, we thought of utilizing the annotated file given in xml format. We planned to train our model using images containing single objects. So, initially our focus was to get cropped image of all objects contained within a single image.

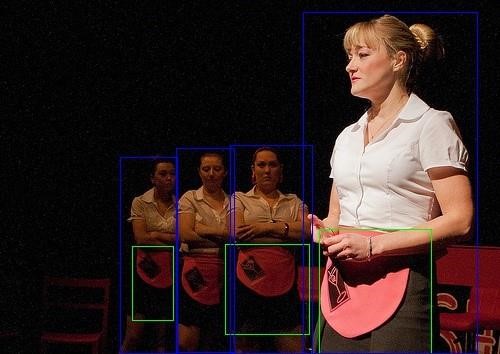
From annotation file of any image, we extracted image file database, folder name, filename and features of each object like xmin (minimum x co-ordinate), xmax (maximum x co-ordinate), ymin (minimum y co-ordinate), ymax (maximum y co-ordinate) and its unique name.

For each individual object in each image, we created a separate image and stored it in a folder. For training purpose, these images containing a single object are used.

**4.2 Example demonstrating Cropped Images**

****

**Fig. 4: Original Image containing multiple objects.**



**Fig. 5: Segmented Image according to the co-ordinates given in xml format image containing multiple objects**



1. (b) (c)



(d) (e) (f) (g)

**Fig. 6**

**Above 7 images are the cropped images containing a single object each which can be given as an input to the training model**

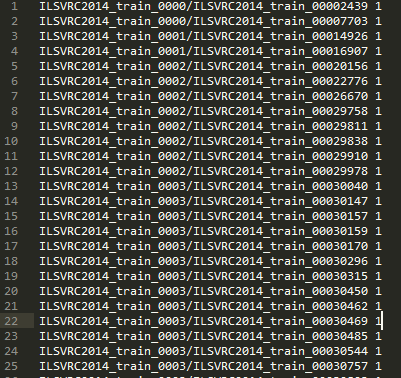
**4.3 Further Preprocessing**

The ImageSets folder of our training dataset contains 200 text files each for different category. Each text file contains paths of images belonging to that particular category. These images are spanned across various folders.

We worked on classification of one category of images as there are 450,000 images in total across 200 categories. For category 1, there are around 63,727 images on which we worked upon.

**Preprocessing Steps:**

1. The text file is read from ILSVRC/ImageSets/DET is converted to a list.



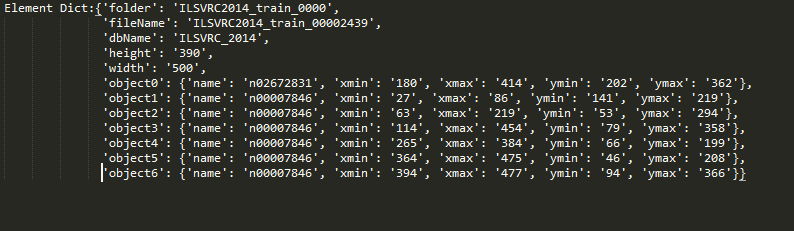
**Fig. 7 Text file containing list of image paths and its annotation path**

1. Based on the image and its annotation path, annotation file (XML formatted file) is read and its data is stored in a python dictionary.



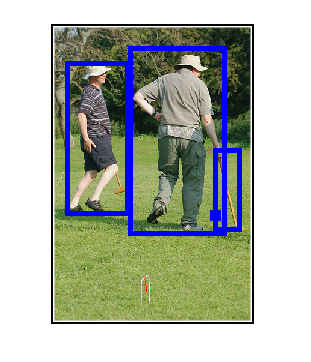
**Fig. 8 XML formatted annotation file of an image**

1. This dictionary has image name, width, height and object details of each object.



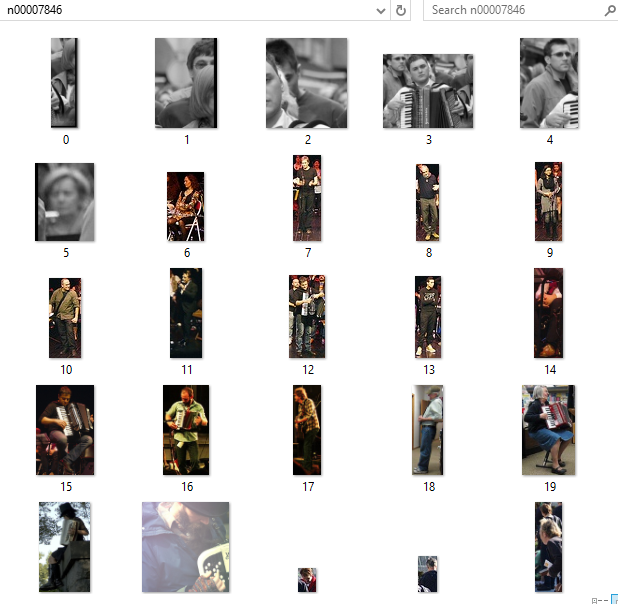
**Fig. 9 Dictionary containing image path, name height, width and details of every object within that image**

1. Based on path of image, it is read and objects are detected based on the annotation file. For all detected objects, separate images are created.

****

**Fig. 10 Original image Fig. 11 Image with object boundary**

1. We used hold out validation approach for splitting the images of category 1 into training and testing.
2. We created different directories for each object class. Each cropped image is stored in its corresponding class directory. For example, the object\_id n00007846 represents ‘Human’ so a directory with name ‘n00007846’ will be created which will contain all cropped images of human.

****

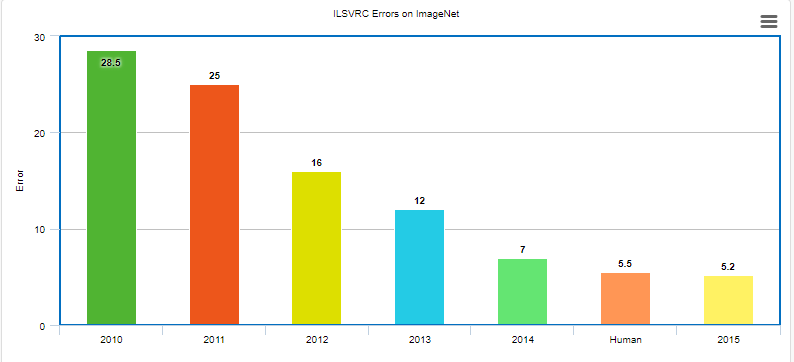
**Fig. 12 Directory containing images of human**

**Chapter 5**

**Proposed Solution and Methods**

**5.1 Convolutional Neural Networks**

Bar chart shown below shows the statistics of ILSVRC Errors on ImageNet by Convolutional Neural Networks and humans. It can be observed that over the years from 2010 to 2014, there has been a drastic decrease in error using Convolutional Neural Networks. In 2015, the error observed is less in case of Convolutional Neural Networks compared to humans. This

****

**Fig. 13 Comparison of ILSVRC Errors by CNN and humans on ImageNet**

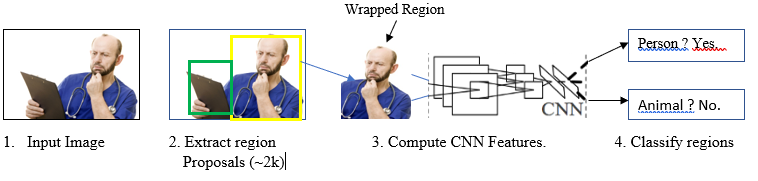
Convolutional Neural Networks are very powerful for image classification task. In a simple neural network, every image pixel is connected to every neuron which makes the network less accurate and computationally expensive. For any given image, two image pixels that are closer to each other are more correlated in comparison to those that are far away from each other. By avoiding such unnecessary connections, convolution resolves this issue. CNN responds to small portion of complete visual field.

**5.2 Regional Convolutional Neural Networks**

Convolutional Neural Networks cannot be used for complicated tasks. For an image containing multiple overlapping objects, Convolutional Neural Networks fails as it can’t detect boundaries around objects.

**Inputs for R-CNN - Image**

**Outputs for R-CNN – Bounding boxes + labels for each object in the image**

****

**Fig. 14 R-CNN: Regions with CNN features**

Regional Neural Network propose a bunch of bounding boxes in the images and checks whether those boxes correspond to actual object or not. Regional Neural Network creates bounding boxes using a technique called Selective Search. Selective Search scans the through varied size windows. For each size, Selective Search groups adjacent pixels together on basis of color, aspects, texture or intensity for object identification.

Regional Neural Networks adds a Support Vector Machine (SVM) which aids in detecting whether it is an object or not and if it is an object, it helps in identifying it.

**Bounding Boxes Improvement**

After proposal of boundaries around objects, the final step of Regional Convolutional Neural Networks is to run a simple linear regression to find the most possible tight boundary around objects.

**Inputs for R-CNN - sub-portions of the images which corresponds to objects**

**Outputs for R-CNN - New bounding boxes**

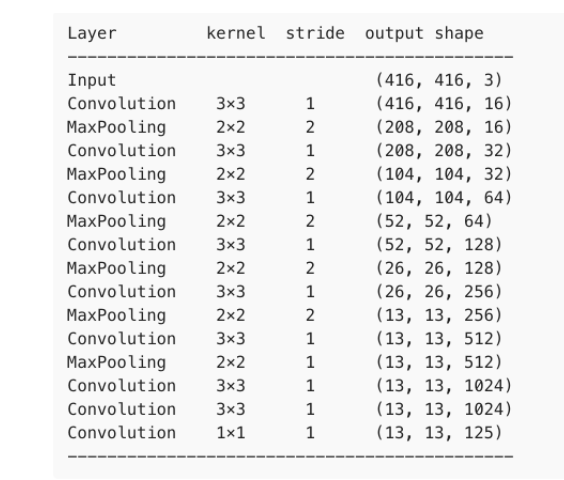
**5.3 MATLAB**

We explored various toolboxes of MATLAB like Image Processing Toolbox and Neural Network Toolbox for multi-object detection. It’s functioning was appropriate for our task but the toolkit was commercial and was not available for free.

**5.4 Yolo: Real-Time Object Detection**

Yolo stands for You Only Look Once. It is an object detection system used in real time. Yolo uses a total unique approach compared to traditional object detection systems. Other object detection systems repurpose classifiers for detection. Model is applied to an image at manifold scales and locations and score is calculated for every region. The regions that have high score are considered potential detected objects.

Yolo uses a single neural network for entire image. This neural network divides the image into grids and predicts bounding boxes and probabilities corresponding to each region. The boundary boxes are weighted by probability predictions.



**Fig. 15 Architecture of YOLO**

Yolo scans the entire image at test time and thus its predictions are decided by global context in the image. Regional Neural Network need thousands of networks for a single image while Yolo needs only one. Thus, Yolo is 1000 times faster compared to R-CNN and 100 times faster compared to Fast R-CNN.

**Fig. 15 Work flow of Yolo Object Detection System**

We have used pre-trained network of Yolo. Below are the steps we followed:

1. We installed Cython. The pre-trained network which we are using, uses cpp to train the network and predict weights. To make the network compatible with Python, we used Cython.
2. There is one labels.txt file which contains all unique classes.
3. First we loaded the yolo.weights file.

Download link for weights: <https://pjreddie.com/darknet/yolo/>

1. We created a new model and initialized it.
2. We loaded yolo.weights into our newly created model for reusability of convolutional layers.
3. We trained the network using newly initialized model and training images.