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

3.4 Data Distribution

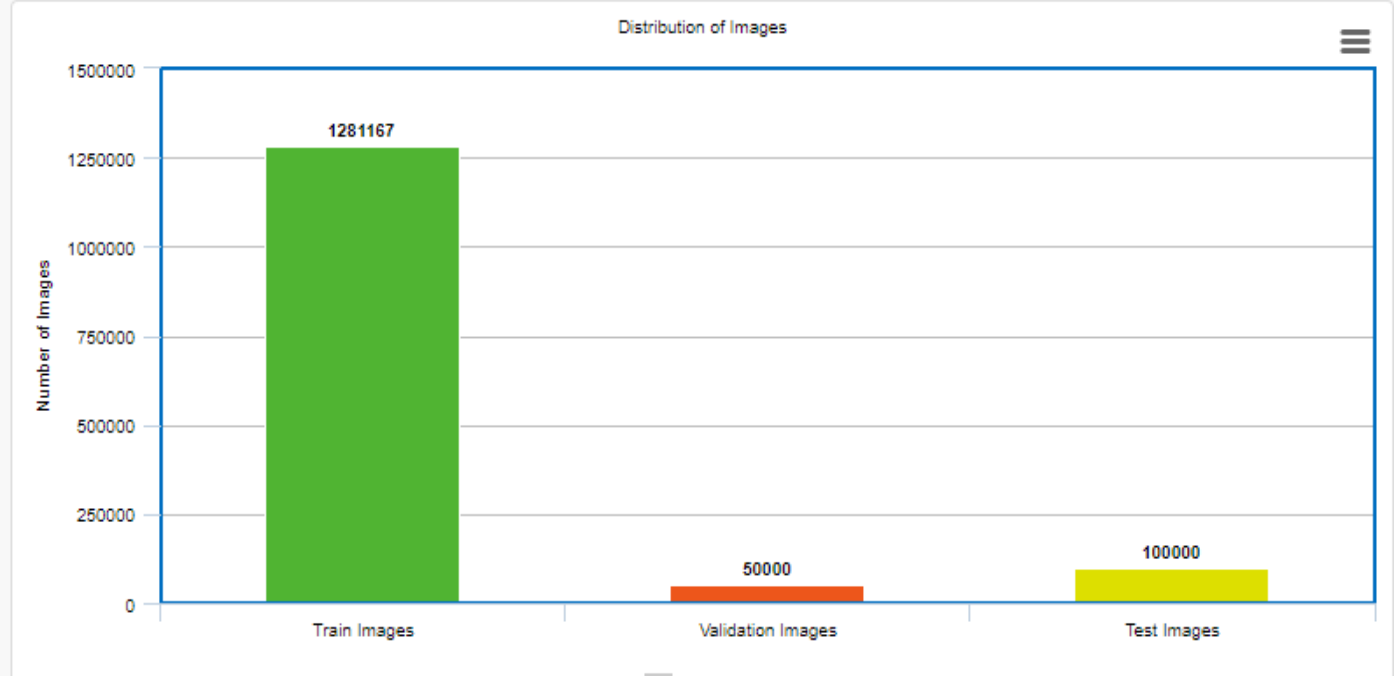

Fig. 1 Distribution of images across training, validation and testing dataset

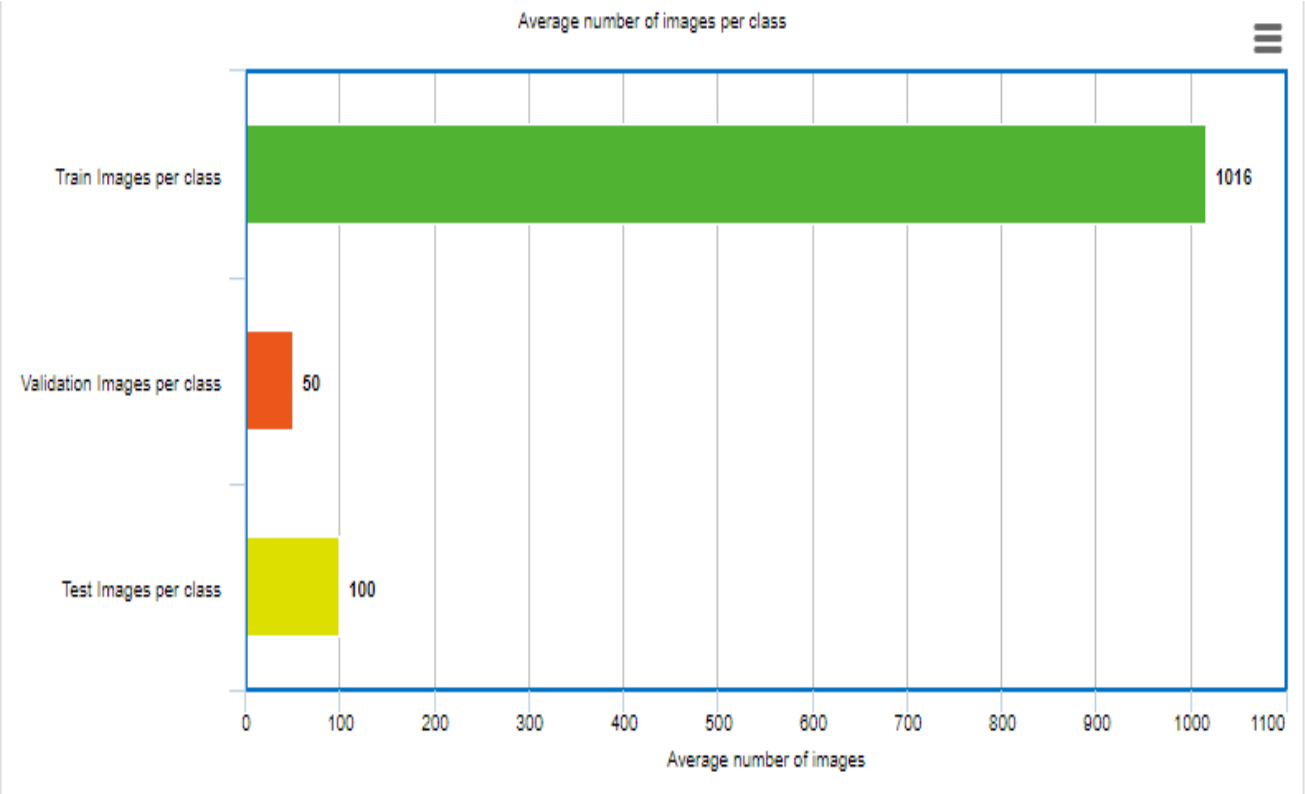

Fig. 2 Average number of images per class across training, validation and testing datasets

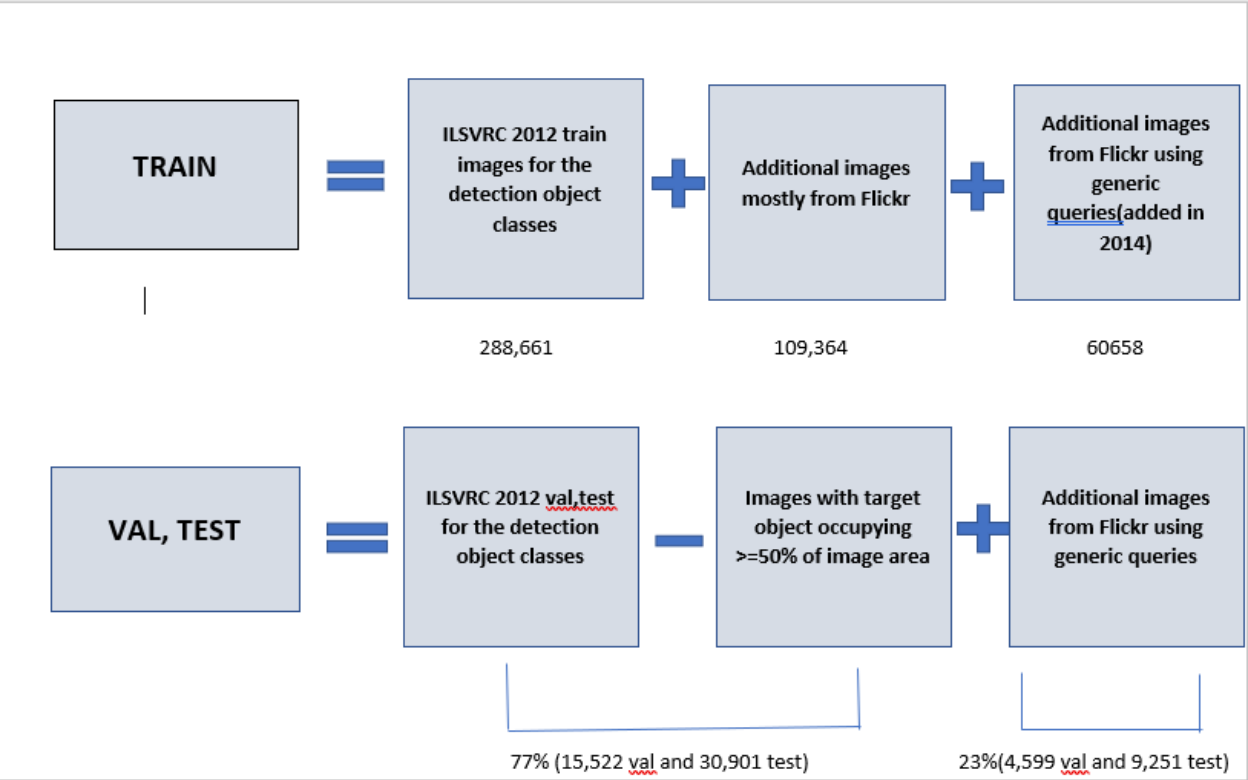

Fig. 3 Summary of images collected for the Object Detection Task

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.

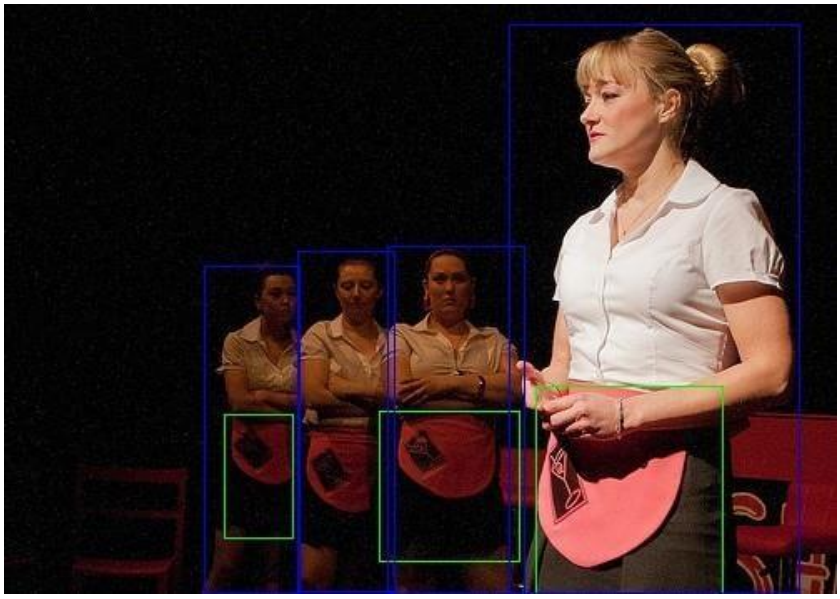

Fig. 5: Segmented Image according to the co-ordinates given in xml format image containing multiple objects

(a) (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.

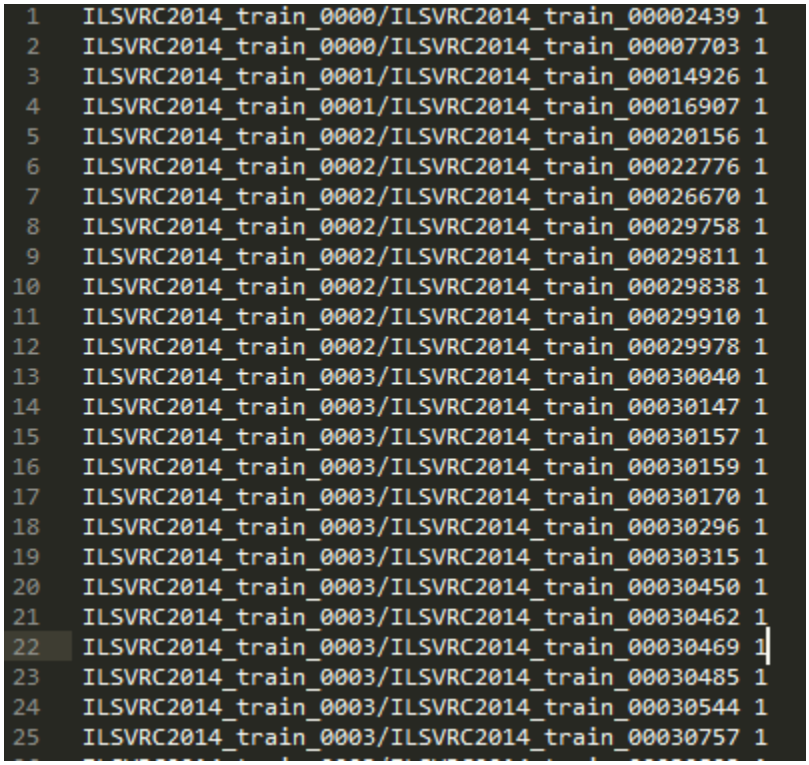

Fig. 7 Text file containing list of image paths and its annotation path

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

```
<annotation>
    <folder>ILSVRC2014_train_0000</folder>
    <filename>ILSVRC2014_train_00000001</filename>
    <source>
        <database>ILSVRC_2014</database>
    </source>
    <size>
        <width>500</width>
        <height>354</height>
    </size>
    <object>
        <name>n03770439</name>
        <bndbox>
            <xmin>132</xmin>
            <xmax>173</xmax>
            <ymin>246</ymin>
            <ymax>320</ymax>
        </bndbox>
    </object>
    <object>
        <name>n03770439</name>
        <bndbox>
            <xmin>225</xmin>
            <xmax>309</xmax>
            <ymin>244</ymin>
            <ymax>334</ymax>
        </bndbox>
    </object>
    <object>
        <name>n03770439</name>
        <bndbox>
            <xmin>319</xmin>
            <xmax>431</xmax>
            <ymin>229</ymin>
            <ymax>353</ymax>
        </bndbox>
    </object>
    <object>
```

Fig. 8 XML formatted annotation file of an image

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

```
Element Dict:{'folder': 'ILSVRC2014_train_0000',
              'fileName': 'ILSVRC2014_train_00002439',
              'dbName': 'ILSVRC_2014',
              'height': '390',
              'width': '500',
              'object0': {'name': 'n02672831', 'xmin': '180', 'xmax': '414', 'ymin': '202', 'ymax': '362'},
              'object1': {'name': 'n00007846', 'xmin': '27', 'xmax': '86', 'ymin': '141', 'ymax': '219'},
              'object2': {'name': 'n00007846', 'xmin': '63', 'xmax': '219', 'ymin': '53', 'ymax': '294'},
              'object3': {'name': 'n00007846', 'xmin': '114', 'xmax': '454', 'ymin': '79', 'ymax': '358'},
              'object4': {'name': 'n00007846', 'xmin': '265', 'xmax': '384', 'ymin': '66', 'ymax': '199'},
              'object5': {'name': 'n00007846', 'xmin': '364', 'xmax': '475', 'ymin': '46', 'ymax': '208'},
              'object6': {'name': 'n00007846', 'xmin': '394', 'xmax': '477', 'ymin': '94', 'ymax': '366'}}
```

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

4) 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

5) We used hold out validation approach for splitting the images of category 1 into training and testing.
6) 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.

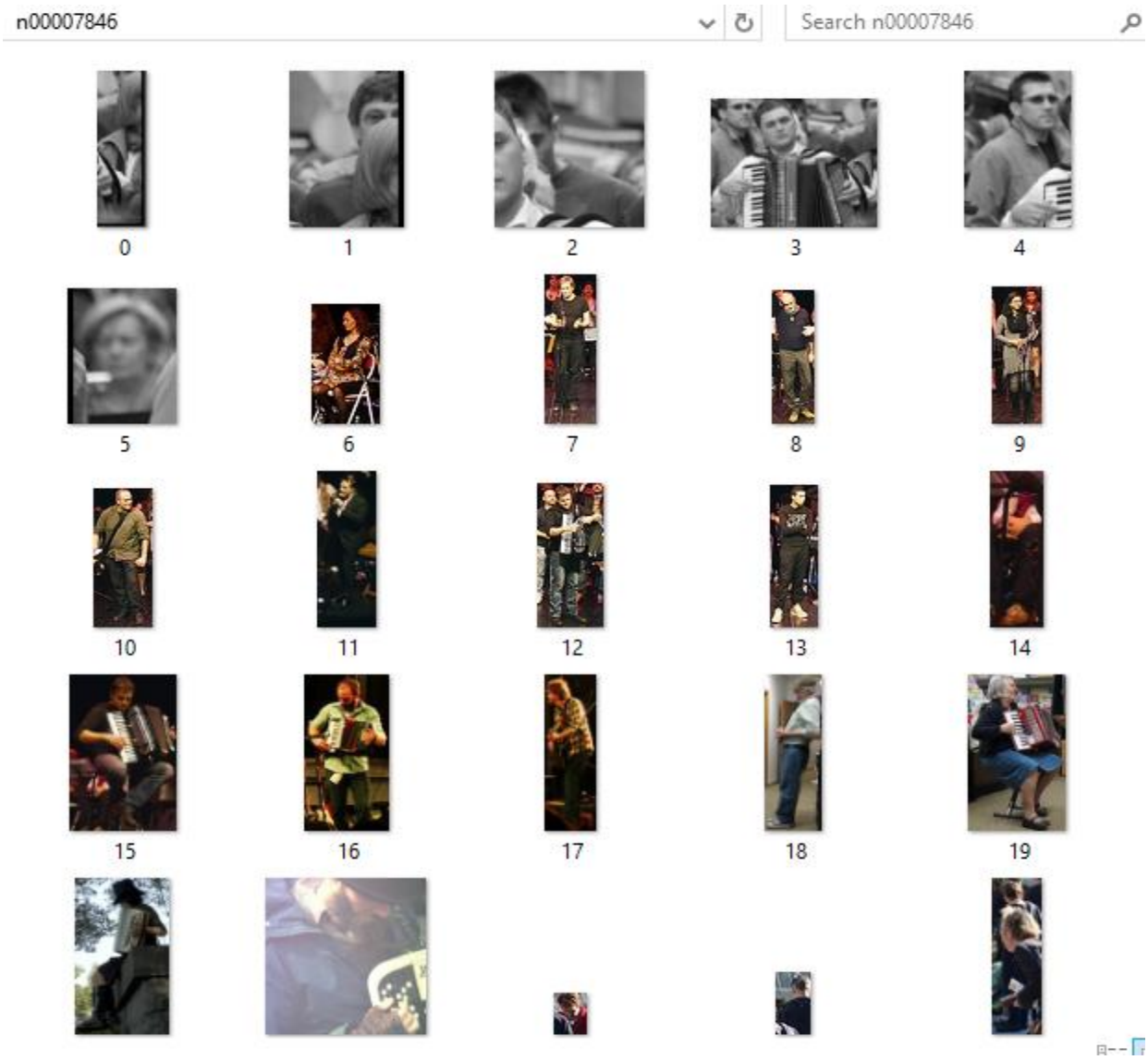

Fig. 12 Directory containing images of human

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

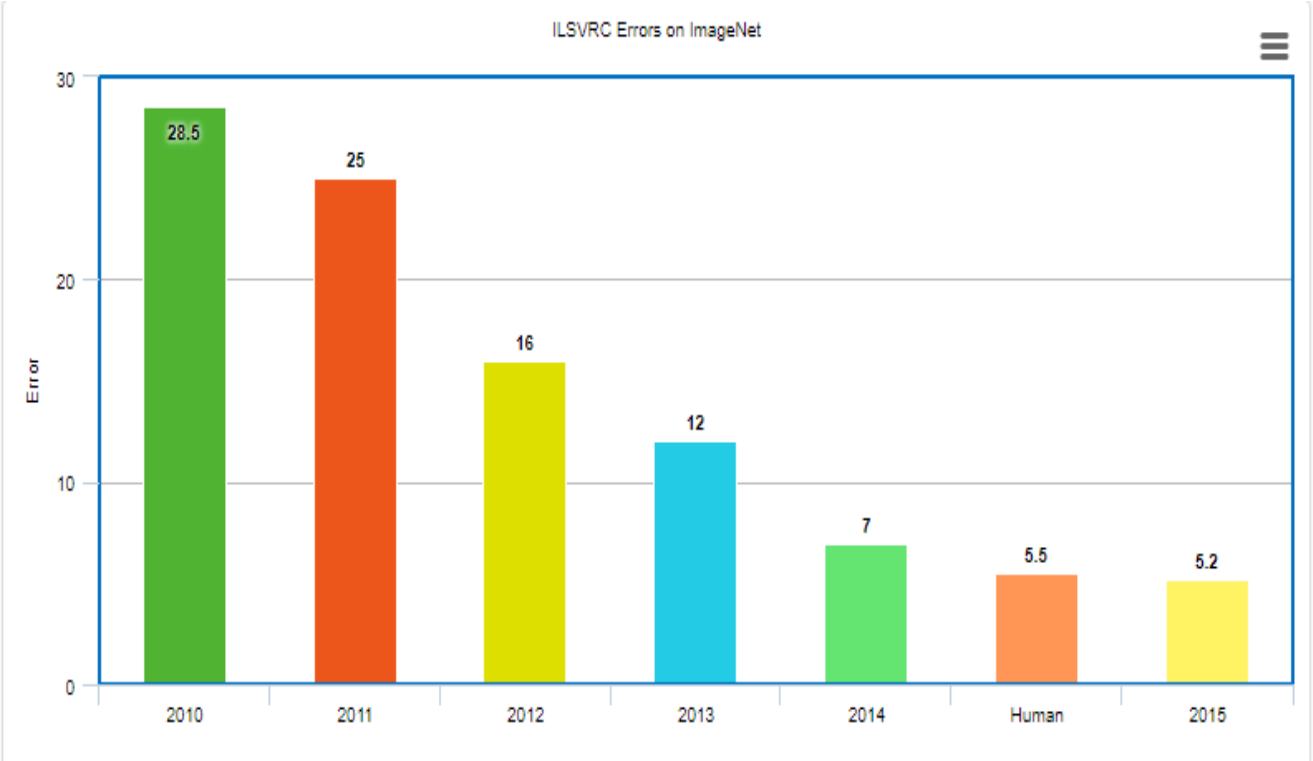

Fig. 13 Comparison of ILSVRC Errors by CNN and humans on ImageNet

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

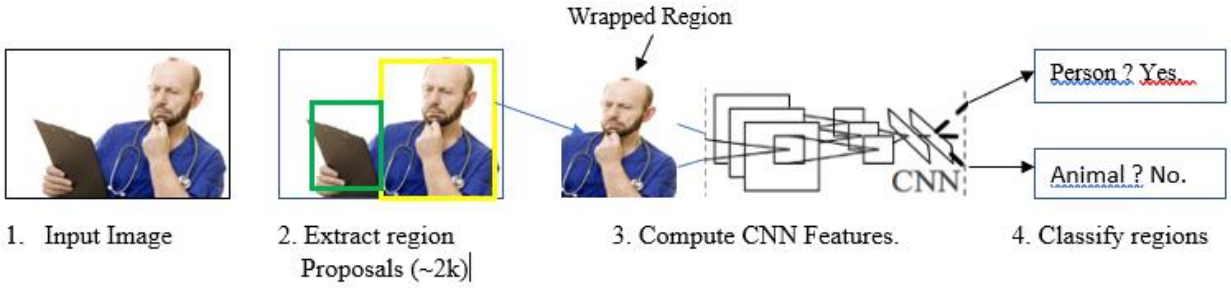

Fig. 14 R-CNN: Regions with CNN features

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.

Layer	kernel	stride	output shape
Input			(416, 416, 3)
Convolution	3×3	1	(416, 416, 16)
MaxPooling	2×2	2	(208, 208, 16)
Convolution	3×3	1	(208, 208, 32)
MaxPooling	2×2	2	(104, 104, 32)
Convolution	3×3	1	(104, 104, 64)
MaxPooling	2×2	2	(52, 52, 64)
Convolution	3×3	1	(52, 52, 128)
MaxPooling	2×2	2	(26, 26, 128)
Convolution	3×3	1	(26, 26, 256)
MaxPooling	2×2	2	(13, 13, 256)
Convolution	3×3	1	(13, 13, 512)
MaxPooling	2×2	1	(13, 13, 512)
Convolution	3×3	1	(13, 13, 1024)
Convolution	3×3	1	(13, 13, 1024)
Convolution	1×1	1	(13, 13, 125)

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.

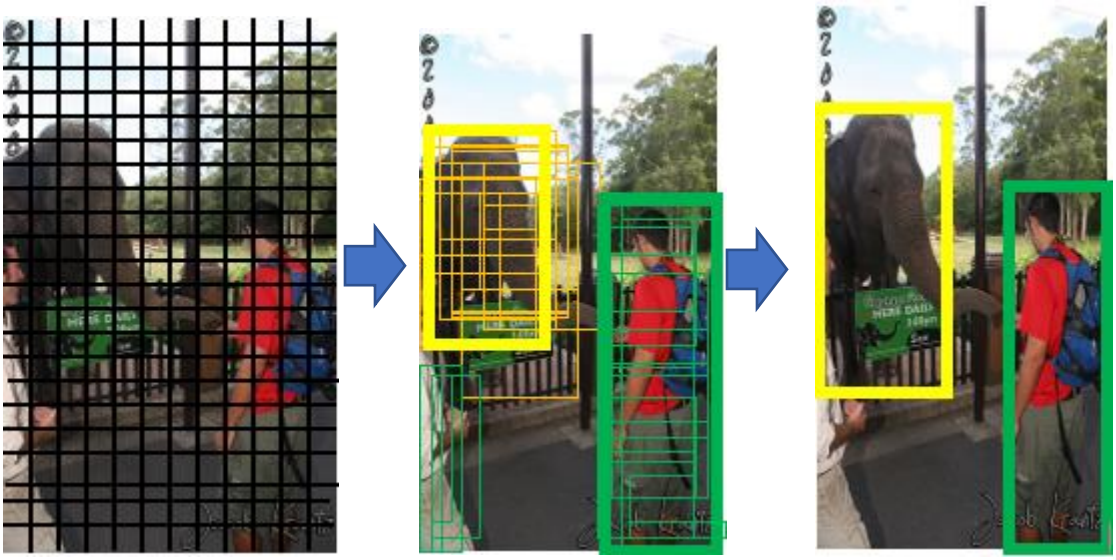

Fig. 15 Work flow of Yolo Object Detection System

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/
4) We created a new model and initialized it.
5) We loaded yolo.weights into our newly created model for reusability of convolutional layers.
6) We trained the network using newly initialized model and training images.

Chapter 6

Experimental Results and Analysis

6.1 Convolutional Neural Networks

6.1.1 Results

```
Epoch 1/50
125/125 [==============================] - 92s 733ms/step - loss: 0.1689 - acc: 0.9760
Epoch 2/50
125/125 [==============================] - 94s 754ms/step - loss: 0.2659 - acc: 0.9835
Epoch 3/50
125/125 [==============================] - 88s 707ms/step - loss: 0.2982 - acc: 0.9815
Epoch 4/50
125/125 [==============================] - 86s 686ms/step - loss: 0.2579 - acc: 0.9840
Epoch 5/50
125/125 [==============================] - 87s 693ms/step - loss: 0.3707 - acc: 0.9770
Epoch 6/50
125/125 [==============================] - 91s 725ms/step - loss: 0.3062 - acc: 0.9810
Epoch 7/50
125/125 [==============================] - 85s 682ms/step - loss: 0.3546 - acc: 0.9780
Epoch 8/50
125/125 [==============================] - 86s 686ms/step - loss: 0.2498 - acc: 0.9845
Epoch 9/50
125/125 [==============================] - 87s 694ms/step - loss: 0.2659 - acc: 0.9835
Epoch 10/50
125/125 [==============================] - 87s 695ms/step - loss: 0.3385 - acc: 0.9790
```

Fig. 15 Results using Convolutional Neural Networks for first 10 epochs

```
Epoch 41/50
125/125 [==============================] - 90s 719ms/step - loss: 0.3304 - acc: 0.9795 -
Epoch 42/50
125/125 [==============================] - 89s 711ms/step - loss: 0.2821 - acc: 0.9825 -
Epoch 43/50
125/125 [==============================] - 89s 715ms/step - loss: 0.3062 - acc: 0.9810 -
Epoch 44/50
125/125 [==============================] - 92s 737ms/step - loss: 0.2095 - acc: 0.9870 -
Epoch 45/50
125/125 [==============================] - 91s 729ms/step - loss: 0.2659 - acc: 0.9835 -
Epoch 46/50
125/125 [==============================] - 102s 818ms/step - loss: 0.3626 - acc: 0.9775
Epoch 47/50
125/125 [==============================] - 98s 781ms/step - loss: 0.2982 - acc: 0.9815 -
Epoch 48/50
125/125 [==============================] - 101s 807ms/step - loss: 0.2901 - acc: 0.9820
Epoch 49/50
125/125 [==============================] - 99s 789ms/step - loss: 0.3224 - acc: 0.9800 -
Epoch 50/50
125/125 [==============================] - 99s 793ms/step - loss: 0.2579 - acc: 0.9840 -
```

Fig. 16 Results using Convolutional Neural Networks for last 10 epochs

6.1.2 Analysis

Convolutional Neural Network works well for images having single object. But for real world, there are complex scenarios. There are complicated sights, overlapping objects having varied background. For such images, we need to detect boundaries, build bounding boxes to proceed further. So, it can be asserted that Convolutional Neural Networks are not enough for ImageNet Object Detection Challenge.

6.2 Yolo: Real-Time Object Detection

6.2.1 Results

Fig. 17 Original Image

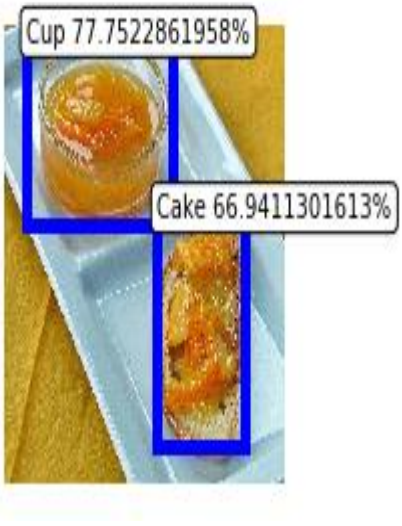

Fig. 18 Corrected Predicted annotated Image

Fig. 19 Original Image

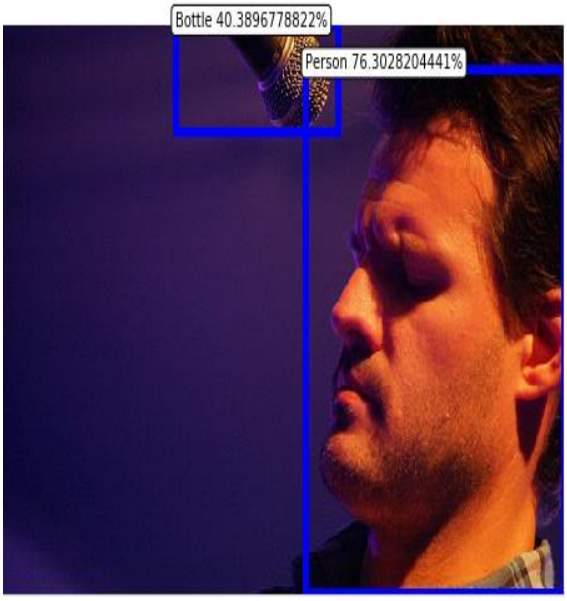

Fig. 20 Incorrected Predicted annotated Image

```
Accuracy 67.27597713470459
Precision 0.6
```

Fig. 21 Evaluation measures Accuracy and Precision for about 65k images

6.2.2 Analysis

By using Yolo, we got 67.27% accuracy and 0.6 precision on over 65K images. We observed that for many images the bounding boxes are correctly formed, and the objects are correctly predicted. While for some of them, objects are correctly detected but they are not recognized correctly.

Chapter 7

Future Scope and Extensions

ImageNet Object Detection Challenge is a Kaggle competition allocated for research of 12 years. Thus, it is a challenging task which involves a great deal of research. At present, we worked on around 65K images belonging to one category. We tried and analyzed techniques like Convolutional Neural Network, Regional Convolutional Neural Network, MATLAB toolboxes and Yolo- Real Time Object Detection technique and obtained moderate results. But we are going to extend our project on a large scale by working on entire dataset with the help of tools like Google Cloud Service or Amazon Web Services to handle large quantity of data. We will explore more methods and ways to improve performance of our task.

Chapter 8

Conclusion

We attempted multiple solutions for the problem of ImageNet Object Detection Challenge and observed that this problem is really challenging. We researched various approaches like Convolutional Neural Networks(CNN), Regional Convolutional Neural Networks(R-CNN) and Yolo: Real Time Object Detection technique. Convolutional Neural Network performs well for images comprising single objects but fails in case of images with multiple objects having different orientations, texture, inter-overlapping and many other characteristics. Due to this drawback of CNN, we proceeded with more research and found one enhanced version of it i.e. R-CNN. R-CNN provides good advantages for multi-object detection. But it requires thousands of networks for a single image unlike Yolo which can fulfil this task with a single network. This makes Yolo 1000 times speedy compared to R-CNN. We opted for Yolo technique finally and used their pre-trained weights to train our model. Yolo detects individual objects by forming boundaries around them. For each boundary formed, it calculates confidence measure based on probability. We tested for around 5000 images and got 67.27% accuracy. We observed that Yolo succeeds in detecting bounding boxes but fails in correct predictions of some objects.

Chapter 9
Contribution of Team Members

Preprocessing

1) Extracting cropped images containing one single object from the original image.

    - Parsing xml formatted annotation file to retrieve image path, size, dimensions.

        - **Nidhi Vaishnav**

    - Image extraction from annotation details like xmin, xmax, ymin, ymax which are the co-ordinates of the rectangle containing the individual class object using OpenCV

        - **Maitri Shah**

2) Creating different directories for each object type to segregate images belonging to different classes.

    - There is one text file which contains paths and annotation file paths for around 65K images.
    - Using hold-out approach to segregate training and testing as 90% and 10% divisions respectively.

        - **Shivani Thakkar**

    - Parsing this text file and extracting cropped images using the logic applied in preprocessing task 1.

        - **Harsh Bhalani**

Model Creation and Results generation

1) Research about different models that can be applied for General Image Object Detection and evaluating their feasibility for our task.
    - **Harsh Bhalani**
    - **Shivani Thakkar**
    - **Maitri Shah**
    - **Nidhi Vaishnav**

2) Convolutional Neural Network (CNN)

    - Model creation and training the model

        - **Maitri Shah**

    - Hyperparameter tuning and finding validation accuracy

        - **Nidhi Vaishnav**

    - Predicting the results from the model

        - **Shivani Thakkar**

    - Concluding that CNN is not appropriate enough to accomplish our task.

        - **Harsh Bhalani**

3) Study regarding limitations of CNN for multi-object image classification.
   Finding alternate ways like,
    - Regional Convolutional Neural Network (R-CNN) which overcomes limitations of CNN by detecting bounding boxes surrounding all individual objects.
    - MATLAB Neural Network Toolbox and Image Processing Toolbox.

        - **Nidhi Vaishnav**
        - **Maitri Shah**

4) The better way to reduce complexity of image detection, YOLO (You Only Look Once) object detection as it uses only single network for entire image which makes it computationally less expensive in terms of time compared to R-CNN.

- Pre- trained model has been used for transfer learning which adds the first layer of weights.
- Created a new model through this pre-trained model on our dataset.
- Predicted the class objects with the confidence measure associated with each of them.
- Another measures for evaluation of the model used are Accuracy and Precision.

    - **Harsh Bhalani**
    - **Shivani Thakkar**

5) Report and documentation

    - **Shivani Thakkar**
    - **Maitri Shah**
    - **Harsh Bhalani**
    - **Nidhi Vaishnav**

References:

http://image-net.org/challenges/LSVRC/2014/

https://elitedatascience.com/beginner-kaggle

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/

https://pjreddie.com/darknet/yolo/

https://www.tensorflow.org/

https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

https://en.wikipedia.org/wiki/Convolutional_neural_network

https://courses.cs.washington.edu/courses/cse590v/14au/cse590v_wk1_rcnn.pdf

http://machinethink.net/blog/object-detection-with-yolo/