Introduction

Looking around whatever we see around and feel around especially when it comes down to devices and the usability, feasibility and dependability of us on those devices, we knowingly or unknowingly are generating data. Lots and lots of data. Gigabytes of data being generated from the apps we visit, what we do on those apps, what are we watching, what are we liking, what we are listening, what are we searching. Each and everything is generating data. Now earlier the data generated as a result was not used mainly due to lesser number of devices generating less data. But in the past decade with the uncontrollable insurgence of data because of readily available devices, higher computational availability, plethora of apps making our lives simple and our dependability on getting things done through them.

From the very first time when Stuart Russell and Peter Norvig wrote the famous textbook "*Artificial Intelligence: A Modern Approach"* [1] explaining and combining introductory and advanced topic and concepts of artificial intelligence [1] together in quite some depth explaining and providing information about the historical implications and backdrops of various algorithms clearly defining and drawing the line between what artificial intelligence (in short AI) and machine learning (in short ML) comprise of. When skimming thorough the book, you can find every buzzword currently we are dealing with in the field of modern machine learning.

Now exploring the field of modern machine learning, let's look in the current 3 most wide open field in machine learning. They are

- Supervised Machine Learning
- Unsupervised Machine Learning
- Reinforcement Learning

Now let's dig a little deeper into these different areas and paradigms of machine learning.

Let's start with "Supervised Machine Learning" [2]. Supervised machine learning [2] as the name suggests is a type of machine learning that uses human's to give supervision to the task it is asked to perform. The task can be a regression task (i.e. fitting a line to a bunch of points) or prediction tasks where you are given a bunch of random examples and you have to teach your machine to predict on unknown samples of data, or it can be classification task where you teach the machine to classify something based on various examples you give. A few can be Bayesian Networks [2], Logistic Regression [2], Decision Trees [2], Random Forests [2], SVM [2], k-NN [2].

Now having covered a basic introduction on one of the techniques, let's cover another technique namely "Unsupervised Machine Learning". Now given the name one can assume what it means and probably you will be correct in your assumption. Unsupervised Machine Learning [3] algorithm as the name suggests works without any supervision or in sense we are not explicitly telling our machine learning algorithm about what to look for. We are just giving it a bunch of points to figure it out on it’s own and give us some sort of result based on it’s observation. The one area where this sort of approach can be used is if we don’t know what is the main target variable. Some of the approaches as mentioned and outlined in the paper given by [3] broadly classifying it into multiple facets such as Hierarchical Learning [3] which contains deep learning and artificial neural networks [3] , data clustering containing Bayesian, hierarchical clustering. Dimensionality reduction [3] is also one of the approaches that fall into the unsupervised learning paradigm. Similarly in the sense outlier detection can also be classified in the area of unsupervised machine learning. In the area containing unsupervised neural networks [3], there are a lot of bifurcations like autoencoders [5], Self Organizing Maps [6], Convolutional Neural networks [4,7] etc. So in practical applications it seems like the area of unsupervised machine learning has the upper hand mostly because of the reason that most data in this world is complex and lacks structure.

Now after a brief introduction into the field of supervised and unsupervised learning, one of the most interesting field of machine learning comes in the form reinforcement learning. A type of learning that is based on the approach of how we humans and animals act in real life. Everything one does is a culmination of cause and effect of certain actions taken in certain states leading to certain output based on the motivation of taking that action. In the same way when some action and state is given to a computer, it tries to learn which state is the best. Which action to take in a particular state so that one can reach another state, in return getting some reward for taking that specific action. One of the most referenced work in the field of reinforcement learning comes from the work of Sutton and Barto [8], the pioneers in the field of reinforcement learning and publishing the holy bible of reinforcement learning academia in a book named “Reinforcement Learning: An Introduction” [8].

The authors in their works have given a complete collection of what all led to the creation of this fascinating field of reinforcement learning. In recent years reinforcement learning has come up as one of the biggest areas of research with companies like Deepmind, Meta investing heavily in this field. The algorithmic idea behind reinforcement learning comes in the form of Markov Decision process named after Russian Mathematician Andrey Markov [9]. It defines a mathematical structure to model a process of making decisions in situations where the decision maker tends to influence the output of the situation partly. So whenever it down to some sort of decision making where the actions, states and the outputs tend to follow a sequential flow of data, reinforcement learning has found a opening into the vast world of AI and machine leaning.

Reinforcement learning has become a part of time series analysis where the stock prices and the movements are being studied as they tend to follow a pattern and are more or less sequentially linked to each other. Reinforcement learning has also found it’s footing in the huge area of robotics be it in logistics, be it in healthcare, agriculture and much more. Reinforcement learning has found a breakthrough in the field of computer vision with the integration of Deep learning and Q-Learning in the famous paper “Human-Level control through deep reinforcement learning” [10].

Now coming to the part of object detection as the main theme of the work that will be integrated with the previous work the literature survey in the upcoming pages covers the work that has been done in the past and the way where the area of object detection is headed and what it is going to look for in the future.

Literature Survey

Object detection can be defined as a process in which the specific architecture or approach tend to extract features from the images, learn them and then try to localize the object out of all the various object of interests. There is a slight misnomer when it comes up regarding Object detection. Object detection is just a way to realize the object present in an input image with a bounding box, on the contrary in image classification, the aim is to just classify if an object is present in the image or not and give a prediction in form of a probabilistic number.

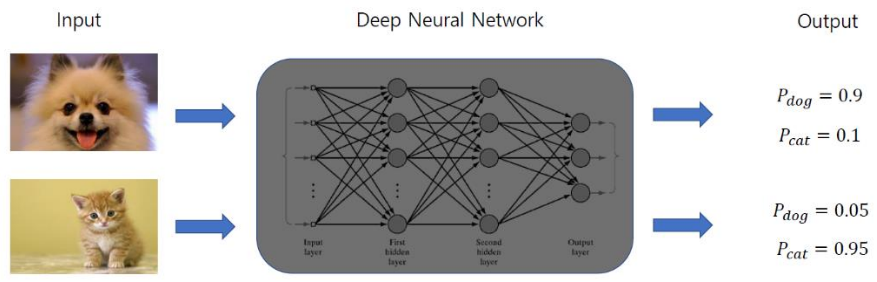

Fig 1: An example of Image Classification (binary) [11]

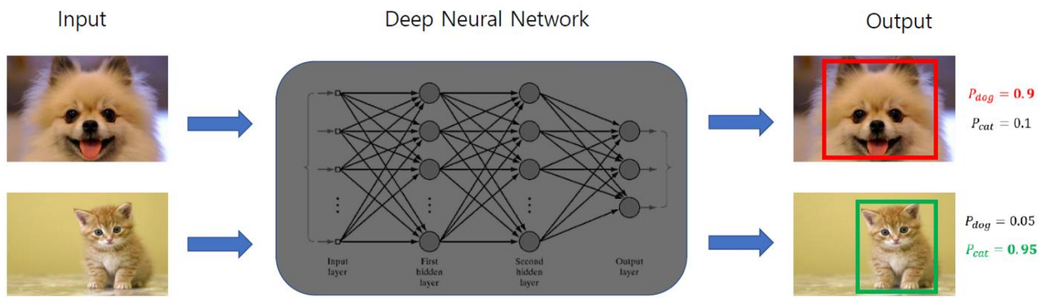

Fig 2: Object Detection (Classification + Localization) [11]

One can look at an Object Detection problem in a sense in which two operations are happening at the same time. One is that you need to classify whether the object is present in the image or not a.k.a image classification, but also with that we need to predict a bounding box around the object of interest that can be modelled as a regression problem. Now it's not necessary that an image dataset will contain images with just one or two object of interests. When it comes to real world applications of object detections be it in a supply chain company, logistics, self driving cars, drones, manufacturing, multiple objects with varied classes can come up. So it is necessary to take in to consideration that objects belonging to many classes can be a part of an image at the same time. We can divide the problem as:

- Multiple label classification
- Finding the bounding box in terms of its coordinates like **x, y, w, h** where x and y defining the coordinated of the center of the image and w and h defining the heights and width of the image.

Object Localization

The main objective of object localization is to predict the object in an image as well as its boundaries. The difference between object localization and object detection is subtle. Simply, object localization aims to locate the main (or most visible) object in an image while object detection tries to find out all the objects and their boundaries.
An image classification or image recognition model simply detects the probability of an object in an image. In opposite to this, object localization refers to identifying the location of an object in the image. An object localization algorithm will output the coordinates of the location of an object with respect to the image. In computer vision, the most popular way to localize an object in an image is to represent its location with the help of bounding boxes.
To measure the performance of an object detection algorithm Intersection Over Union otherwise known as IoU or even Jaccard-Index is considered widely in the deep learning community.

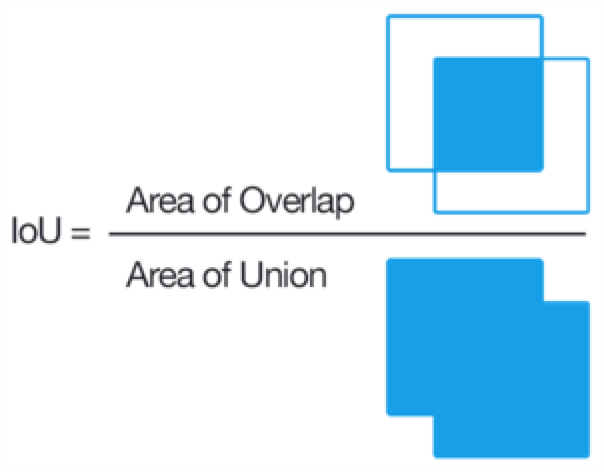

Fig 3: Intersection Over Union

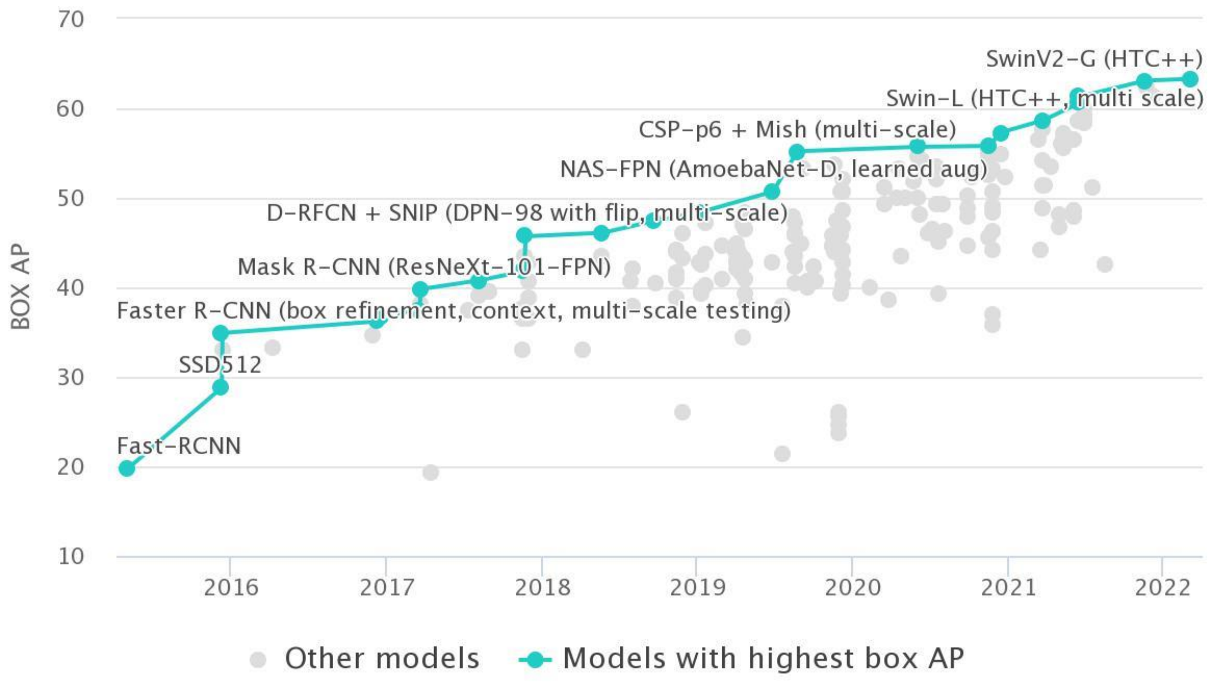

Fig 4: Increase in box AP over the years on COCO Dataset [12]

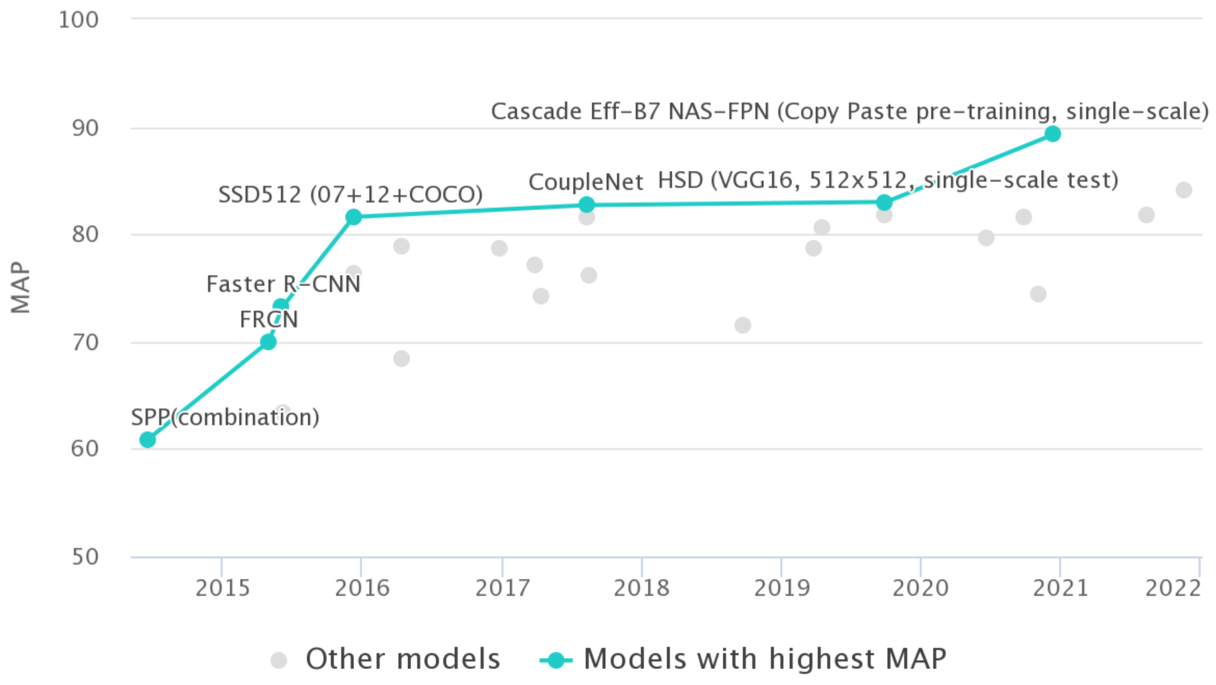

Fig 5: Increase in mAP for PASCAL VOC Dataset [13]

RCNN [14]:

R-CNN [14] is an acronym for Region-based Convolutional Neural Network. The main idea behind the multiple versions of the RCNN family is detecting objects using region proposals. Region proposals are a method that was introduced to the world which was used to localize objects within an image.

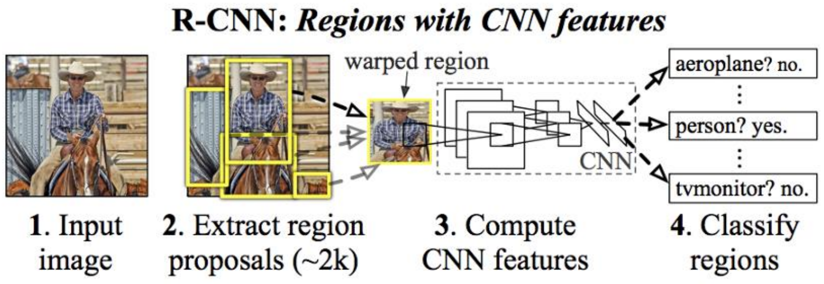

Fig 6: R-CNN approach [14]

As can be seen in the image above before passing an image through a network, we need to extract region proposals or regions of interest using an algorithm such as selective search. Then, we need to resize (wrap) all the extracted crops and pass them through a network. Finally, a network assigns a category from C + 1, including the ‘background’ label, categories for a given crop. Additionally, it predicts delta X_s and Y_s to shape a given crop.

Extracting region proposals [14]:

Selective Search is a region proposal algorithm used for object localization that groups regions together based on their pixel intensities. So, it groups pixels based on the hierarchical grouping of similar pixels. In the original paper, the authors extract about 2,000 proposals.

Positive vs. negative examples:

After one extracts the region proposal, one has to also label them for training. Therefore, the authors label all the proposals having IoU of at least 0.5 with any of the ground-truth bounding boxes with their corresponding classes. However, all other region proposals that have an IoU of less than 0.3 were labelled as background. The rest of the proposals were simply ignored.

$$t_x = (G_x - P_x)/P_w \qquad (6)$$
$$t_y = (G_y - P_y)/P_h \qquad (7)$$
$$t_w = \log(G_w/P_w) \qquad (8)$$
$$t_h = \log(G_h/P_h). \qquad (9)$$

Fig 7: Bounding Box Regression [14]

The image above shows deltas that are to be predicted by CNN. So, x, y are center coordinates. And w, h are width and height respectively. Finally, G and P stand for ground-truth bounding box and region proposal respectively. It is important to note that the bounding box loss is only calculated for positive samples.

Loss:
The total loss is calculated as the sum of classification and regression losses. However, there is a coefficient lambda for the latter one, which is 1,000 in the original paper. Note that the regression loss is ignored for negative examples.

Fast RCNN [15]:
The biggest roadblocks of what the R-CNN [14] approach showed were:

- First being the multi-stage implementation and expensive training: RCNN [14] required separate training processes for roughly all the stages of the network that contained fine-tuning a CNN on object proposals, feature vectors that were generated from each proposals using CNN which were then used to make a SVM lean to classify and learning a bounding box regressor to fine-tune the object proposals. These multiple implementation stages proved to be a burden in view of time, computation and resources. Like to train the SVM, one would need the features of thousands of possible region proposals to be written to the disk from the previous stage.
- Second being the slow test time: Given such a multi-stage pipeline, detection even using a simple CNN architecture like VGG [16] took about 47 seconds per image.

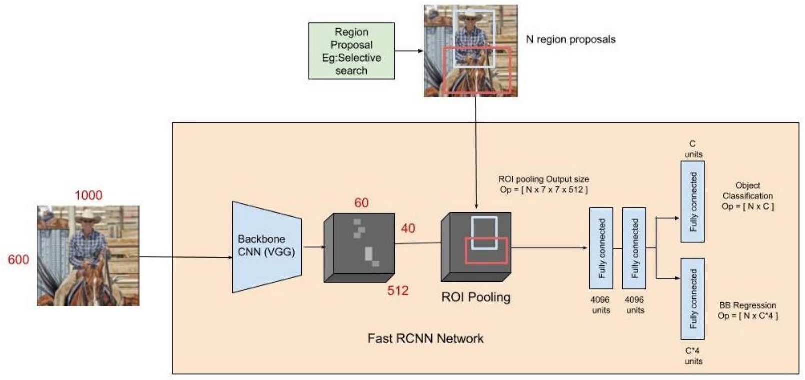
Region Proposal Eg:Selective search
N region proposals
1000
600
Backbone CNN (VGG)
60
40
512
ROI Pooling
ROI pooling Output size Op = [N x 7 x 7 x 512]
Fully connected
4096 units
4096 units
C units
C*4 units
Object Classification Op = [N x C]
BB Regression Op = [N x C*4]
Fast RCNN Network

Fig 8: Fast RCNN Approach using VGG [16] backbone

Implementation detail of Fast-RCNN [15] Architecture:
To have a better understanding of the why-how of the reason behind the improved efficiency of Fast R-CNN when compared to the performance of R-CNN , a deeper exploration of the architecture is a necessity.

- Fast R-CNN consists of a CNN (usually pre-trained on the ImageNet classification task) with its final pooling layer replaced by an “ROI pooling” layer and its final FC layer is replaced by two branches —
  - a (K + 1) category softmax layer branch and
  - a category-specific bounding box regression branch.

- The entire image is fed into the backbone CNN and the features from the last convolution layer are obtained. Depending on the backbone CNN used, the output feature maps are much smaller than the original image size. This depends on the stride of the backbone CNN, which is usually 16 in the case of a VGG backbone.
- Meanwhile, the object proposal windows are obtained from a region proposal algorithm like selective search. Object proposals are nothing more than just rectangular regions on the image that specify an objects presence in the image.
- The portion of the backbone feature map that belongs to this window is then fed into the ROI Pooling layer.
- The ROI pooling layer is a special case of the spatial pyramid pooling (SPP) [17] layer with just one pyramid level. The main work that the pooling layer does is that it divides the features from the selected proposal windows (generated by the region proposal algorithm) into smaller sub-windows of size *h/H* by *w/W* and performs a pooling operation in each of these sub-windows. Due to this a fixed-size output feature map of size (H x W) irrespective of the input size is generated. H and W are chosen in such a way that the output is compatible with the network's first fully-connected layer. The chosen values of H and W in the Fast R-CNN paper is 7. Like regular pooling, ROI pooling is carried out in every channel individually.
- The output features from the ROI Pooling layer (N x 7 x 7 x 512 where N is the number of proposals) are then fed into the successive FC layers, and the softmax and bounding-box-regression branches. The softmax classification branch produces probability values of each ROI belonging to K categories and one catch-all background category. The boundingbox regression branch output is used to make the bounding boxes from the region proposal algorithm more precise.

This led to the following:

- Fast R-CNN achieved state of the art performance at the time of its publication on VOC12, VOC10(with additional data) and VOC07.
- Fast R-CNN processes images 45x faster than R-CNN at test time and 9x faster at train time. It also trains 2.7x faster and runs test images 7x faster than SPP-Net. On further using truncated SVD, the detection time of the network is reduced by more than 30% with just a 0.3 drop in mAP.
- Other contributions of the paper show that for larger networks fine-tuning the conv layers as well (and not just the FC layers like SPPnets [17]) is really important to achieve significantly better mAP. For smaller networks, this improvement might not be as large.
- Multi-task training is not only easier but also improves performance because the tasks influence each other while training. Hence training the different stages of the network altogether improves it's shared representative power (the backbone CNN)

Faster RCNN [18]

In the R-CNN family of papers, the change between versions were usually in terms of computational efficiency, reducing the amount of testing time, and performance improvement in terms of mAP (PASCALVOC). RCNN networks generally were built of :

a) A region proposal algorithm which will generate the bounding boxes for the object present in the image

b) A pretrained feature extraction network which usually is a CNN;

c) A classification layer to predict which class this object belongs to

d) A regression layer to correct the bounding box co-ordinates more precise.

In both the previous iterations namely RCNN [14] and Fast-RCNN [15], region proposal algorithm was the one which took the bulk of computation. R-CNN [14] and Fast R-CNN [15] both used CPU intensive algorithms for proposing regions like Selective search algorithm. The main aim of Faster R-CNN [18] is to do this using another convolutional network known as Region Proposal Networks to generate the region proposals. RPN [18] not only helps in cutting down the time requirements per image but also allows a possible sharing of layers between the RPN [18] and detection network.in-turn improving the total representation of features. Thus in other words "Faster R-CNN" is a detection pipeline that uses the RPN as a region proposal algorithm, and Fast R-CNN [15] as a detector network.

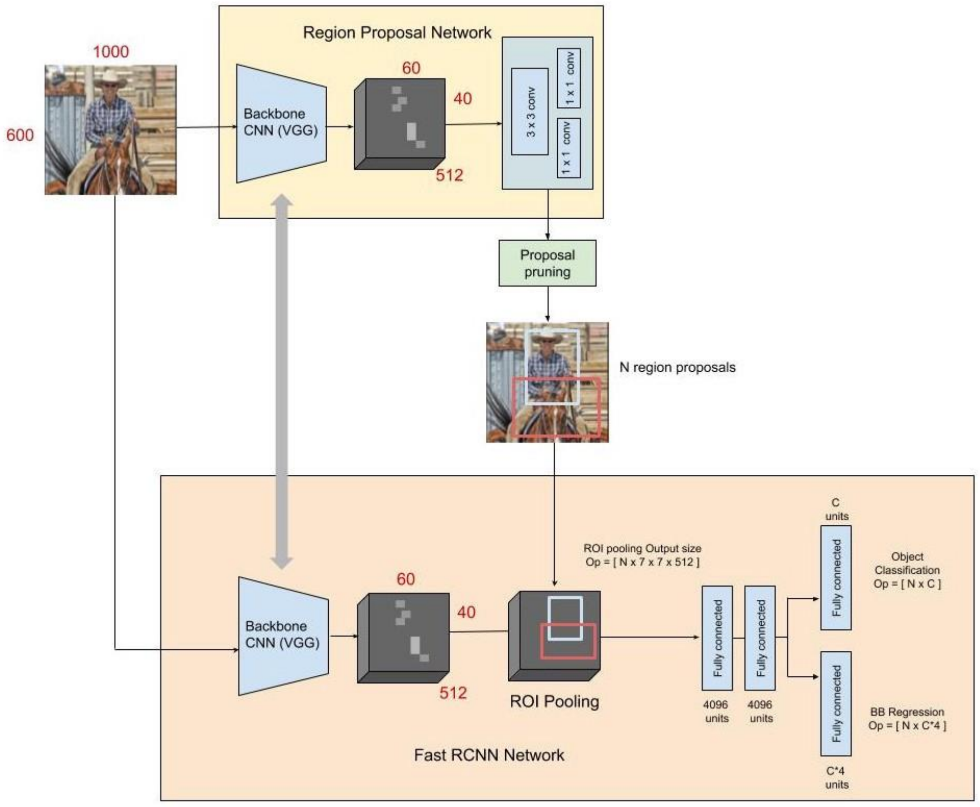

Fig 9: The RPN for region proposals and Fast R-CNN as a detector in the Faster R-CNN detection pipeline

Problem Statement

So regarding the implementation of Faster RCNN on local machine I used a number of various approaches and took help from many github repositories and was finally able to get a similar but a lower number mAP number as mentioned in the original python implementation on Github by one of the authors of Faster RCNN (https://github.com/rbgirshick/py-faster-rcnn).

My machine uses a 4GB Nvidia GTX1650, 8GB RAM, Windows 10, Pycharm Community and Tensorflow 2.3 with Python 3.7. The biggest issue that I faced while implementing the code was the Out-Of-Memory Error. So to circumvent that the implementation used a dense connected network of 1024 neurons each and connected to different number of fully connected layers with a dropout of 0.5. In current work VGG-16 is used as the backbone architecture.

Work Done Till Date

To get a respectable mAP value on the PASCAL VOC 2007 dataset, multiple approaches have been tested out. All the loss outputs and corresponding values of mAP are documented.

1) Using Pretrained Imagenet and Shuffle ON, 10 Epochs, Learning rate= 1e-7 after 6 Epochs:

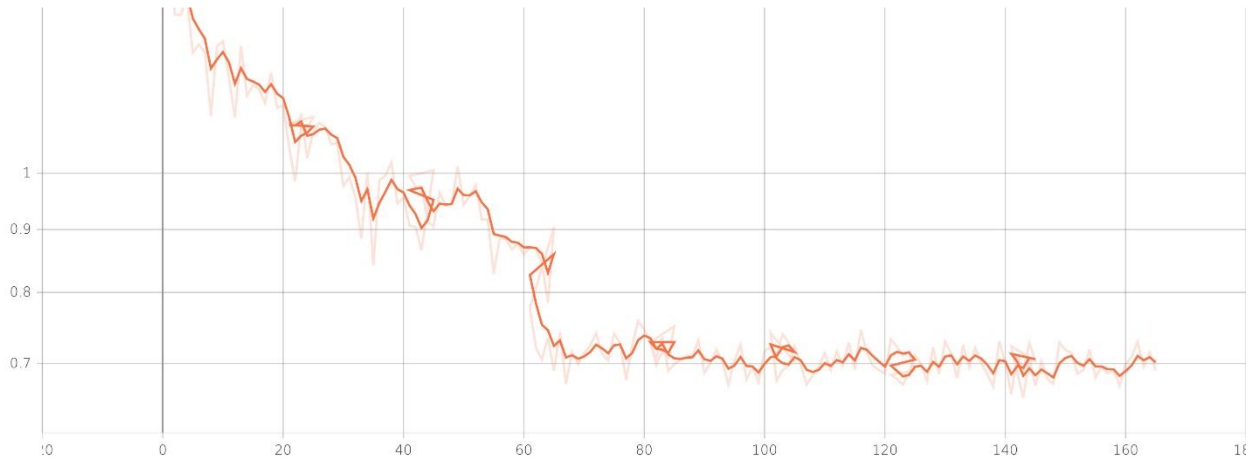

Fig: Total Loss (Loss Value = 0.7012)

mAP (PASCAL VOC 2007) = 34.69%

2) Using Pretrained Imagenet and Shuffle OFF, 10 Epochs, Learning rate= 1e-7 after 6 Epochs:

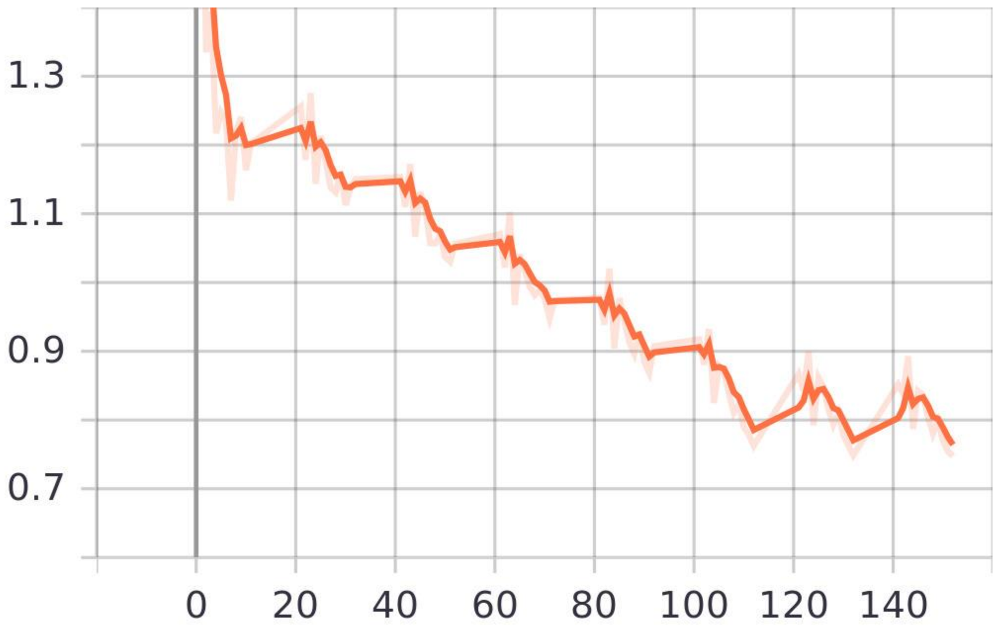

Fig: Total Loss (Loss Value = 0.7325)

mAP (PASCAL VOC 2007) = 34.85%

3) Using Pretrained Unfreezed Conv1 and Conv2, 3 Fully Connected, 10 Epochs, Learning rate= 1e-7 after 6 Epochs:

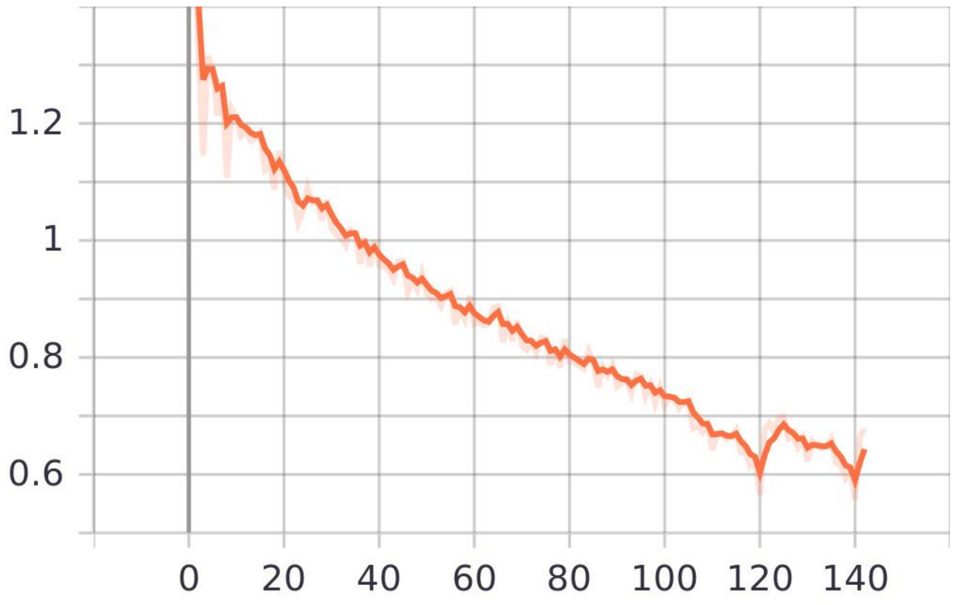

Fig: Total Loss (Loss Value = 0.6481)

mAP (PASCAL VOC 2007) = 43.22%

4) Using Pretrained Unfreezed Conv1 and Conv2, 3 Fully Connected, 12 Epochs, Learning rate= 1e-7 after 6 Epochs:

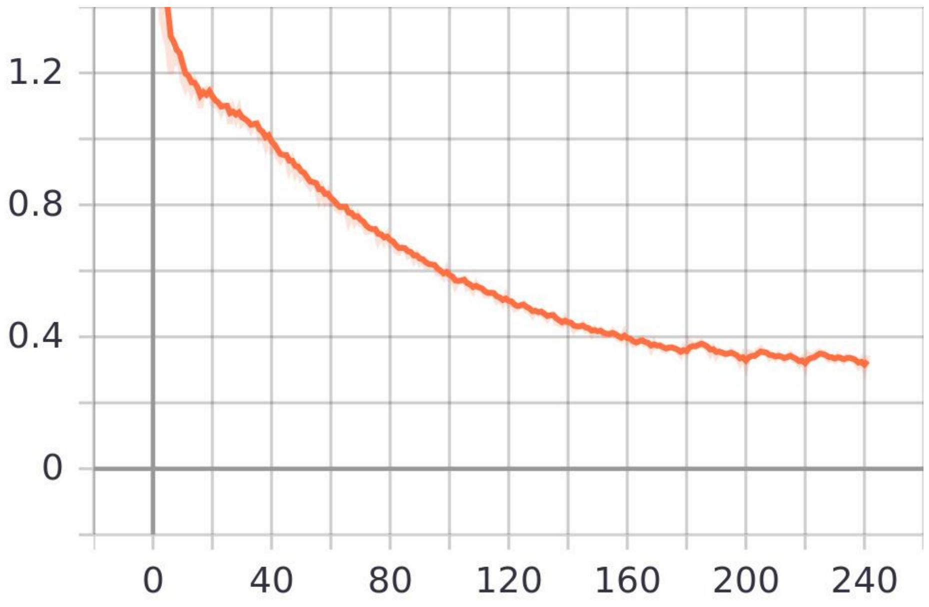

Fig : Total Loss (Total Loss=0.3312)

mAP = 62.41%

Future Work

For future work the following things are planned and to be executed:

1) Using the above trained model predict on “Person” class of PASCAL VOC 2007 Dataset
2) “Person” class being used as an alternative to overlapping objects.
3) Using the predictions from the Faster RCNN Implementation, and based on the ground truth values to be used from PASCAL VOC dataset, find the corrected coordinated of “Person” class in image.
4) Calculate AP value before and after correction.
5) Alternate architecture of DQN network with a different state of the art architecture for feature extraction.

References

[1] Russell, Stuart, and Peter Norvig. "Artificial intelligence: a modern approach." (2002). Textbook, Prentice Hall Publishers.

[2] A. Singh, N. Thakur and A. Sharma, "A review of supervised machine learning algorithms," 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016, pp. 1310-1315.

[3] M. Usama et al., "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges," in IEEE Access, vol. 7, pp. 65579-65615, 2019, doi: 10.1109/ACCESS.2019.2916648.

[4] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436–444 (2015).

[5] Bank, Dor, Noam Koenigstein, and Raja Giryes. "Autoencoders." *arXiv preprint arXiv:2003.05991* (2020).

[6] Kohonen, Teuvo. "Essentials of the self-organizing map." *Neural networks* 37 (2013): 52-65.

[7] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)* (pp. 1-6). IEEE.

[8] Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[9] Puterman, Martin L. "Markov decision processes." *Handbooks in operations research and management science* 2 (1990): 331-434.

[10] Mnih, V., Kavukcuoglu, K., Silver, D. *et al.* Human-level control through deep reinforcement learning. *Nature* **518,** 529–533 (2015). https://doi.org/10.1038/nature14236

[11] Blog Post : What is Object Detection ? (https://medium.com/ml-research-lab/what-is-object-detection-51f9d872ece7)

[12] Paper Collection: On COCO Dataset (https://paperswithcode.com/sota/object-detection-on-coco)

[13] Paper Collection: On PASCAL VOC Dataset (https://paperswithcode.com/sota/object-detection-on-pascal-voc-2007)

[14] Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587. 2014.

[15] Girshick, Ross. "Fast r-cnn." In *Proceedings of the IEEE international conference on computer vision*, pp. 1440-1448. 2015.

[16] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[17] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial pyramid pooling in deep convolutional networks for visual recognition." *IEEE transactions on pattern analysis and machine intelligence* 37, no. 9 (2015): 1904-1916.

[18] Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." *Advances in neural information processing systems* 28 (2015).