CHAPTER 1

INTRODUCTION

1.1. PRELIMENARY INSIGHT

Since the invention of the wheel, humans have always tried to reduce the load and burden of human effort. After the wheel was invented it opened up venues for transportation which included transporting humans, animals, food items, etc. throughout the world. This led to the discovery of gears leading to the invention of cars and automobiles. Just as the discovery of wheels started the industrial revolution, in the same way, the discovery of the "Difference Machine" by Charles Babbage [1] started the revolution in the field of using computers to automate and calculate large calculations in the field of science, mathematics, business and much more.

The concept of automation led to a curious start in the field of machine learning. "Making machines learn and do things on their own", can be understood as a layman's definition of machine learning. The sole focus of a machine learning system is to learn to automate the learning process [2]. Refining the algorithms and the observation that the machine learns to improve the future prediction over time.

Machine learning in itself is a concept modeled in parts, after the human brain. One of the earliest works done in this field was by Donald Hebb in 1948, in his book titled "The Organization of Behavior" [3]. After this one of the most influential works that kick-started the machine learning field was given by one of the most prominent personalities, Alan M. Turing. "Computing Machinery and Intelligence" is one of the key papers published by him in 1949, raised the question of "Can machines think?" [4].The paper argued that there isn't any argument that can convince us that machines don't have the ability to think like humans. The "Turing Test" designed by Alan Turing himself was the one that came up with the concept of identifying whether the answer given to a specific

question is by a machine or a human being. Fig 1.1 illustrates the timeline of the evolution of the field of machine learning and artificial intelligence.

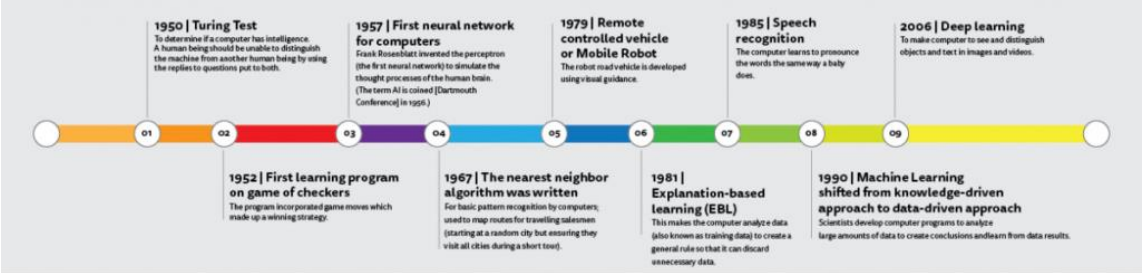

Fig 1.1 Evolution of the field of machine learning and artificial intelligence as we know it to date [5]

In the year 1951, Arthur Samuel from IBM developed a program to play checkers. In the year 1954, John McCarthy, a professor emeritus from Stanford coined the term “artificial intelligence” [6]. Keeping in conjecture to the history of the field of machine learning, in the year 1956, Frank Rosenblatt is credited with the work of the first-ever model of a computational unit modeled exactly like the brain and is best known by the name “Perceptron” [7]. The “Perceptron” can be thought of as the stepping stone for the creation of what we today know as an “Artificial Neural Network”. Moving forward in the year 1966, was the year when Cover and Hart proposed the algorithm famously known nowadays as “K-Nearest Neighbor” [8] which was then proposed to actually find the most efficient route for solving the infamous “Travelling Salesman Problem”. Moving forward in the same era, the creation of multiple layers in the area of neural networks paved a new road for research.

The creation of multiple layers led to the formation of what we today know as a “feedforward neural network”. This decade also is credited with a number of researchers coming up with the idea of one of the most important concepts in the field of deep neural networks known as “backpropagation”. Though the idea of backpropagation has been around for quite a long time, the use of backpropagation as a learning method for the neural networks can be credited to the infamous paper by Geoffrey Hinton named “Learning representation by back-propagating errors” [9]. The concept of backpropagation tells us that any artificial neural network adjusts its layers that are hidden based on reducing the value of a function which it tries to calculate by the difference between the expected and the calculated values which it terms the “error” caused due to mismatch in respective values.

Moving forward machine learning and artificial intelligence went on their separate ways with the latter generally focusing on using an approach based on logic and knowledge which is start different from what machine learning tries to do which is to draw out conclusions based on different algorithms. Machine learning started to leverage the ideas from statistics and probabilities along with concepts from artificial intelligence to solve practical problems and start leveraging for business usage.

1.2. BRIEF SURVEY OF EXISTING LITERATURE

The basic introduction to the field of machine learning was covered in the articles and papers from [1]-[9]. Going in-depth and exploring the various avenues of machine learning like supervised learning, unsupervised learning, and reinforcement learning were covered in the works from [10]-[35]. The works done from [36]-[67] talk about the advancements done in the field of neural networks and how the extension of neural networks in the field of computer vision in the form of convolutional neural networks. The works from [68]-[83] tries to give an in-depth insight into the field of reinforcement learning and the extension to that field in the form of deep reinforcement learning.

1.3. PROBLEM STATEMENT

The major aim of preparing this thesis is to accomplish the following goals:

a) To outline the basics in the field of machine learning and outline the different approaches in the field of machine learning.
b) To do an in-depth study of the field of deep reinforcement learning.
c) To outline the improvement that the experiment did to correct bounding boxes.
d) To provide future scope for the researcher to dive into the fascinating field of deep reinforcement learning

1.4. ORGANIZATION OF THESIS

The complete thesis is organized into multiple chapters as outlined in the following few lines:

Chapter 1: This chapter gives an overview, outlines the content, and contains the literature review that was done when working on the topic.

Chapter 2: This chapter gives a brief introduction to the field of machine learning covering various types and aspects of machine learning.

Chapter 3: This chapter will give a thorough outline of the most used neural network architecture in CNN.

Chapter 4: This chapter will outline the field of reinforcement learning and how it was used in the area of computer vision.

Chapter 5: This chapter will go into the proposed methodology and the working principle of the work that has been done during the complete tenure.

Chapter 6: This chapter will outline the total outcomes that came as a part of different experiments that were performed.

Chapter 7: This chapter will provide a conclusion and the future scope.

CHAPTER 2

MACHINE LEARNING TECHNIQUES

2.1. BACKGROUND

The main problem that machine learning tries to solve is trying to find patterns in the data which act as the fundamental to that data and has a very successful past in doing so. For example, one can quote extensive works done in the field of astronomy back in the 16 century when Kepler was determined to discover some of the hidden patterns on how planets in our solar system tend to revolve around the sun. Similarly one can credit the evolvement of quantum physics at the beginning of the twentieth century to the development and discovery of atomic spectra being regular. Similarly, when it comes to the field of discovering patterns, using various computer algorithms to find out different patterns in the data to discover regularities just like Kepler or like the work done in the field of quantum physics, also after discovering the regularities in the data to help classify data in multiple categories.

Now let's consider one of the simplest examples to outline the work done in the field of pattern recognition, the recognition of handwritten digits. One of the earliest works done in this is can be credited to Yan LeCunn in his work "Backpropagation Applied to Handwritten Zip Code Recognition" [10]. In that, each digit corresponded to 28X28 pixel images which can be converted into a vector of length that consists of 774 numbers. The main aim of this work was to develop a machine-learning algorithm to make it learn to take such vectors of length x which when fed will then produce identification in the term that which of the 10 digits that specific flattened layer of numbers belongs to. Now, this can be categorized as one of the non-trivial problems given the reason that digits can be written in different forms because of the varying handwriting of people writing them.

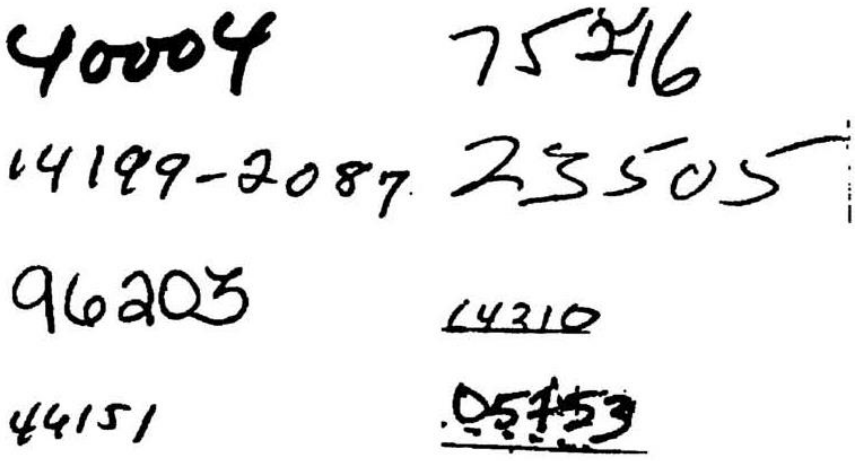

Fig. 2.1 Examples of handwritten digits [10]

So to circumvent the variances that might be due to different sorts of handwriting, some specific heuristics or handcrafted rules can be put in place that tend to determine what digit it can be based on the shape of strokes, but in a practical approach, it is seen that this sort of approach tends to yield bad results.

To improve the result one can use a method that uses a large collection of data commonly known as a *training dataset,* to train the machine learning algorithm by tuning the weights and learnable parameters of such model. Before training the adaptive model, the digits are already segregated based on their corresponding categories in this case every set of numbers is already known to be one of the 10 digits in a decimal number system which is done by individually marking them, in turn taking a lot of human effort. Now after the training dataset is marked there has to be something known as a *target* for the machine learning algorithm to run after, which is a *vector* giving the corresponding digits belonging to every *vector* belonging to the *training dataset.*

The complete execution of the machine learning algorithm can be defined by a function *f(x),* where *x* can be defined as the input given to the machine learning algorithm, which generates an output *f* based on the *target* the machine learning algorithm was given. Now after the algorithm learns what it can base on the training dataset, it then tries to extend its knowledge or the weight adjustments or parameter changes it did to learn the *target,* to more unseen examples, which is known as the ability of *generalization.* In other words, it gives an indication of how a specific machine learning algorithm is able to extend its learning or *generalize* to unknown examples. As compared to all the possible combinations of inputs in the form of input vector *x,* the *training dataset* comprises a very tiny subset of all such inputs so it becomes an absolute necessity for the machine learning algorithm to have the ability to *generalize*.

When it comes to applying various machine learning algorithms in practical cases, one of the steps that are generally taken before feeding in the input to the machine learning algorithm is *pre-processing*. These are generally done to transform the hypothesis space of the input variable making it much easier for the pattern recognition systems to learn the patterns in an even better way. Pre-processing the inputs also tends to convert all of them in the same range which greatly reduces the variability of the data among themselves. Sometimes the pre-processing step taken is also known as *feature extraction.* One of the reasons to do pre-processing is to reduce the computational cost it will take for the pattern recognition algorithm to learn the differences in the input data if they are scaled to the same range.

Generally in the paradigm of machine learning, two types of use cases are presented, one of them being classification and the other being regression. Both classification and regression are methodologies for predicting class labels, with a very small difference among them. When it comes to the classification model, the model tends to predict discrete labels for classes. On the other hand, when using a regression model, the labels to be predicted are continuous in nature.

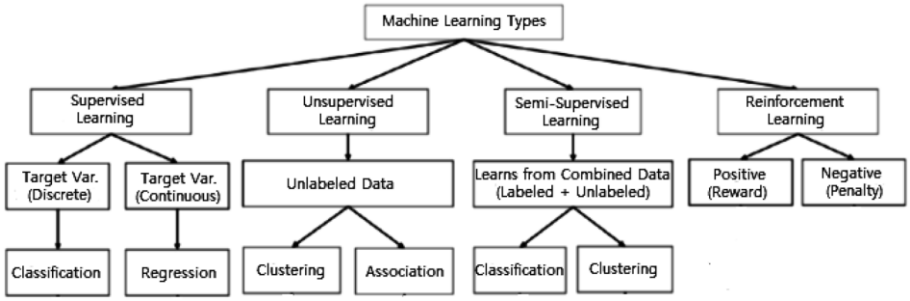

Fig. 2.2 Different types of machine learning algorithms [11]

2.2. SUPERVISED LEARNING

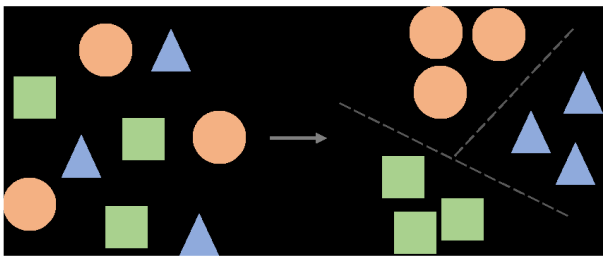

Fig. 2.3 Organizing input dataset into known classes [11]

Supervised learning [12] as the name suggests is a machine learning paradigm that learns by supervision. The term *supervision* means learning with the knowledge that what is right and what is wrong based on the input you have got and based on the output you are expected to give.

The process of learning generally contains two parts, learning or training, and testing. During the process of *training*, the training dataset mentioned in the above text is given as an input from which the pre-processing step extracts the features in other words known as feature extraction [13]. When testing of the model is done, the trained and fin-tuned model uses the *testing dataset* and then predicts the expected target values [13].

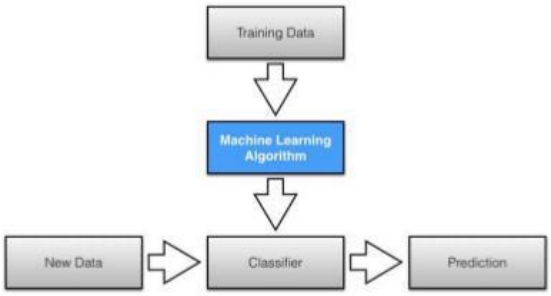

Fig. 2.4 A pictorial representation of a supervised learning process [13]

A supervised learning approach uses a dataset that tends to contain a collection of *features* and the corresponding labels and for the testing part of the algorithm, the *labels* are predicted by the learning algorithm or the *learner*. The aim is to create an estimation system that when given a set of features will be able to predict the corresponding label to the features. The way the prediction system learns and corrects itself is by comparing the output it generates to what the expected outputs are and thus finding corresponding errors. It then rectifies the parameters of the model accordingly.

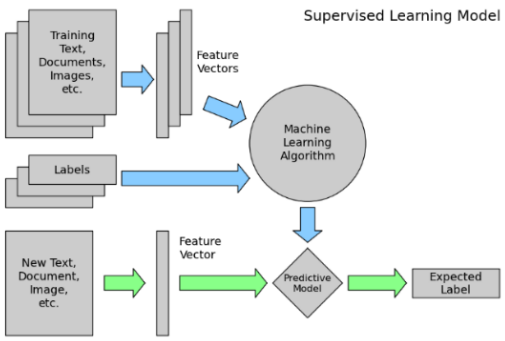

Fig. 2.5 A pictorial representation of a Supervised model [13]

Some of the most used supervised learning [14] algorithms used mainly are for the purpose of classification which includes [15]: Linear classifiers [16], Naïve Bayes Classifier [17], Logistics Regression [18], Perceptron [19] [20], Support Vector Machines [21], Decision Tree [22], Random Forest [23], Artificial Neural Networks [24], Bayesian Networks [25] and much more.

Giving a brief idea about what the above methods use, a linear classifier as the name suggests uses a linear decision boundary to segregate the input data into a number of classes. Also, a linear classifier [26] tends to solve a linear equation that contains variables as the input features and is used when speed is a concern for us. Naïve Bayes [17] uses directed acyclic graphs [26] multiple children nodes and one parent node. The Naïve Bayes [17] classifier inherently assumes that there is independence between the parent nodes and child nodes.

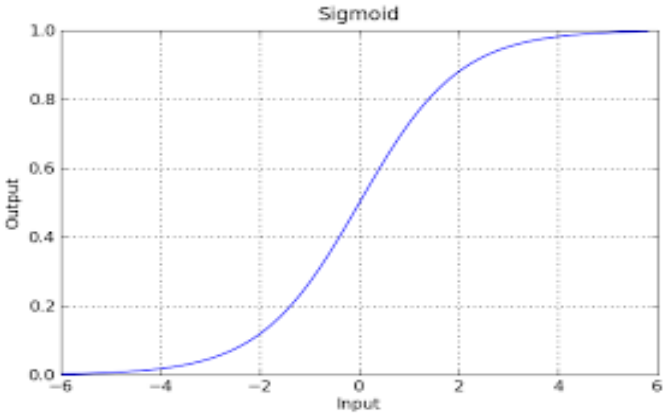

Fig. 2.6 An example of a logistic function [26]

Logistic regression [26] is a classification algorithm that uses a polynomial regression model which uses just one estimator. The classification algorithm lays out the decision boundary when classifying two classes, also providing us with the probability associated with both the classes. Multilayer perceptron differs from neural networks in the sense that [26], the weight updates in a perceptron follow a quadratic problem [26] and have linear constraints [26], in contrast to solving nonlinear functions in neural networks. Support Vector Machines [26] convert a lower-dimensional input data into a feature space with a higher dimension. Then the algorithm tries to find the best separator to differentiate the classes which are drawn as a hyperplane with the closes points to the hyperplane acting as the support vectors for the classification boundary.

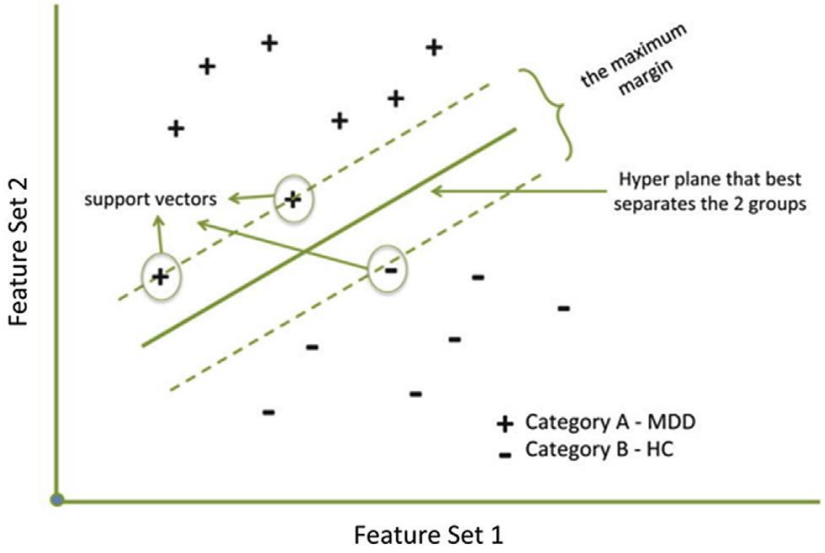

Fig. 2.7 Support Vector Machine [21]

A decision tree [26] in layman's terms tends to follow the "if not this, then this" approach. Nodes in the decision trees compare features among themselves when traversing the length of the tree telling what value a specific node can take. Thus when traversing starts from the root the feature values present in the node then help in classifying based on the node value.

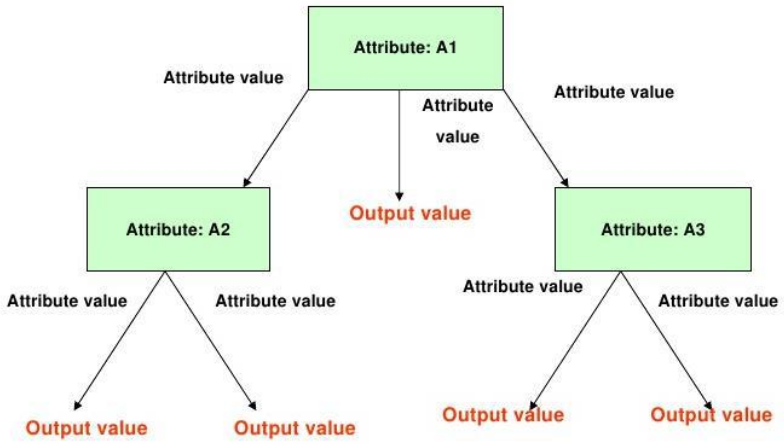

Fig. 2.8 A decision tree approach [26]

2.3. UNSUPERVISED LEARNING

The availability of labelled data sure makes life easier when being used for classification as well as regression. But what happens if we want to classify the given data but no target label is given? That is where unsupervised learning [27] comes into the picture. When we are bound to find hidden patterns which is the main aim of a pattern recognition system, having unlabelled data might cause hindrance, thus unsupervised machine learning models find out patterns on their own which any supervision from the training dataset. In contrast to supervised learning models, these algorithms try to capture the hidden patterns in the dataset, bringing together the data based on the number of similarities and then compressing them for presentation. In the same way, as we differentiate supervised learning in classification and regression, here we bifurcate the unsupervised learning methodology in clustering and association.

When given a collection of unlabelled data, it is assumed that it will have in some sense among them a level of similarity and this is what a clustering algorithm tries to find out. Clustering algorithms tries to find out clusters of data that have the most amount of similarity among themselves and then group them together in one of the clusters. Some of the clustering techniques are K-means clustering [28], K-Nearest Neighbours [29], Hierarchical clustering [30], Principal Component Analysis [31] Singular Value Decomposition [32], Independent Component Analysis [33], and many more.

Giving a brief overview of the aforementioned unsupervised learning techniques, K-Means clustering, is a categorization algorithm that divides a complete dataset into *k* different groups based on the amount of similarity. The similarity is calculated based on different distance-based measurements. We randomly chose k points no the data, calculate the distances and update the coordinates of the corresponding means, keep on running the iteration until and unless all the dataset points belong to one cluster or not. Similarly for KNN, we first select K number of neighbors, then for those K neighbors calculate the corresponding Euclidean distance, taking the K nearest neighbors, we count the number of data points in the category and the one with the maximum number of neighbors that category is assigned to that datapoint.

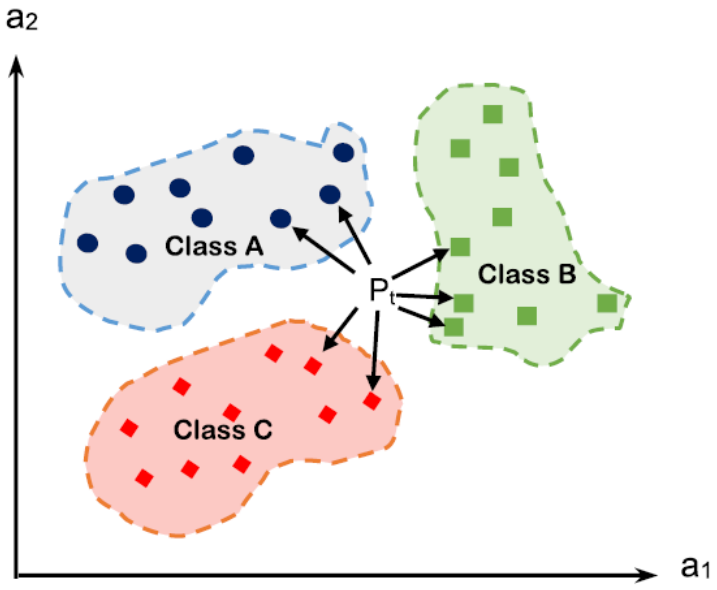

Fig. 2.9 Example of KNN [34]

2.4. REINFORCEMENT LEARNING

One of the types of machine learning that find its origin in psychological texts is reinforcement learning [35]. When considering the basic goals of an artificial intelligence system, automating agents to do work by interaction with the environment and learning the optimal behaviors to accomplish those tasks is one of the prominent ones. So when an experience-driven learning methodology is modelled in a mathematical framework, the resulting methodology is known as reinforcement learning [35]. In general, whenever one needs to solve that can be modelled as action and reward, reinforcement learning is the way to go. Considering an example from real life, when a

kid is growing up and exploring, parents and the environment are the ones who give feedback to the kid whenever the kid does anything that can be anything from playing in a playground and getting hurt to the kid studying for grades in his/her class.

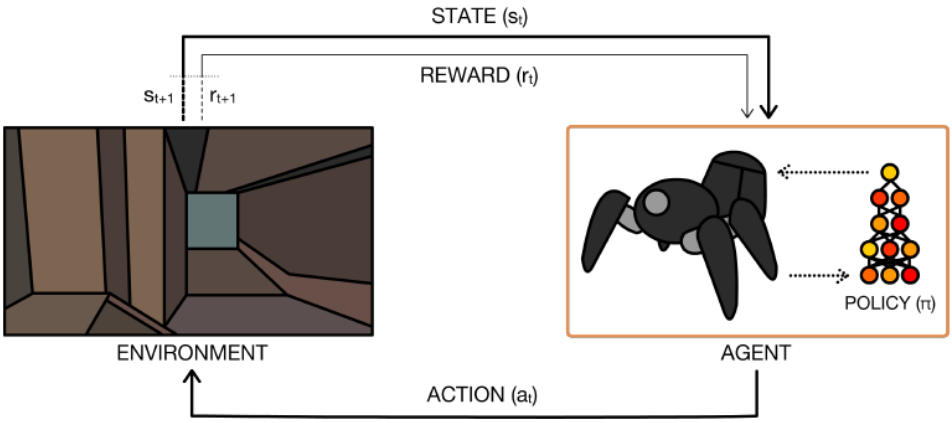

Fig. 2.10 An example of how reinforcement learning works [35]

CHAPTER 3

CONVOLUTIONAL NEURAL NETWORKS

3.1. NEURONS

One of the best examples of models that tend to recognize patterns from things around in the environment is the human mind. The complex biological process that goes on inside the mind of the human is said to be the starting point [36] of the use of creating an artificial form of a biological neuron to find hidden patterns in the environment and data provided, in this case, which would be a collection of numbers whereas in real life the data provided is in multiple ways. It can be an image, someone's voice, some instruction, or any of a hundred other ways. But when trying to simulate a similar way of biological neurons, it was impossible because of the following reasons:

1) We have no idea about the number of neurons that are present in a human brain and how many interconnections are in between them.
2) We have no way to simulate the behaviour of what happens when some sort of operation happens.

Though it is not possible to exactly model what happens in a human neuron, we can somehow use different computation units to create an assumingly close to human neuron model. When talking about some of the interesting features of a normal human neuron and network which is quite superior than any AI system when being used for pattern recognition is:

1) It is way more robust than a neural network because of its high immunity to noisy input.
2) The ability to tolerate fault is higher in the human neural network because of the sheer amount of neurons present inside the brain which is technically not possible to create artificially.
3) In artificial networks, the networks are generally preprogrammed making them less flexible as compared to biological neural networks.

4) The biggest hand that a biological neural network has over a traditional artificial neural network is its ability to compute things in parallel which is not at all possible for artificial neural networks.

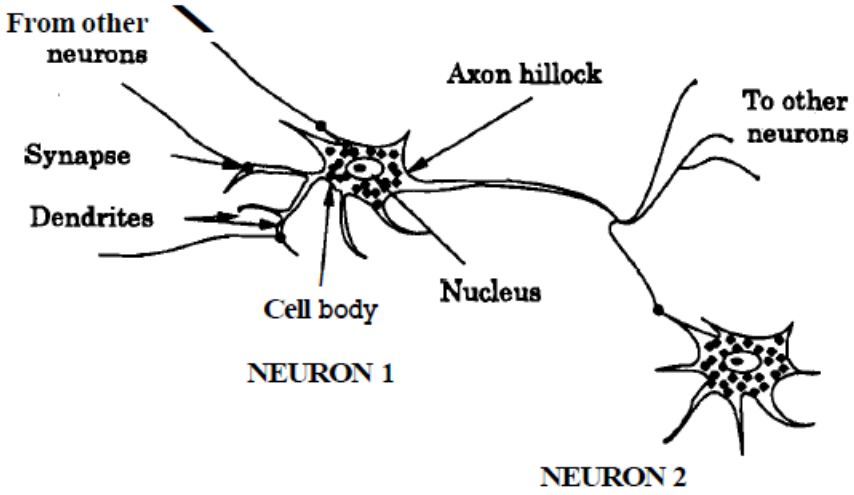

Fig. 3.1 A schematic representation of a biological neuron [37]

The above is a representation of a typical biological neuron. This is what acts as the basic building block of a biological neural network. The cell body, otherwise known as some is the one that encloses the nucleus. The branch-like structure extending from the cell body are known as dendrites. Dendrites are the ones that send and receive signals from other neurons. The longest part of a neuron is known as the axon which also acts as the body of the neuron. The neurons are not connected physically with each other but rather connected with a chemical in between them, which are known as synaptic junctions or simple synapses. The axon ends generally connects with roughly thousands of other neurons which keep on getting bigger and bigger.

The method of transmission is eerily similar to what we see in artificial neural networks. From the sending axon end, a chemical substance is released. This substance increases or lowers the potential inside the receiving cell body. As soon as a specific threshold is reached, the cells generate electrical pulses and the neuron is said to be fired. The synapses are the ones on which chemical reactions take place, whereas the cell body and the axon is the one that deals with the electrical activity. Dendrites are the signal catchers and axons are the transmission channels. Biological neurons are way faster as compared to artificial neurons. They tend to undertake massive parallel processing which is not even in the realm of possibility for an artificial neural network. Biological neural networks are massive and way more complex than any artificial network.

3.2 PERCEPTRON

After the interest in the field of neural networks sparked in the minds of researchers around the globe, the first ones to come up with some sort of mathematical model. They were Warren McCulloch, a neurophysiologist at the University of Illinois at Chicago, and the other, cognitive psychologist Walter Pitts. Both of them together in 1943, published their work "A logical Calculus of the idea Immanent in the Nervous Activity" [38]. They gave a rough idea on how a neuron can be modelled in mathematical terms with the inclusion of things life variable weights, threshold functions, and more. Continuing the works Rosenblatt in the year 1957 came up with the first-ever perceptron model [7]. Rosenblatt was inspired by the Hebbian theory which suggested that brain neurons when undergoing training adapt themselves. This he denoted by the term plasticity. MCP or McColloch & Pitts neurons were defined by the following rules:

1) The output *y* will be binary having values of either 1 i.e. the neuron is firing, or 0 i.e. the neuron is at rest.
2) The excitation is done by *n* number of binary inputs which can take the value of either 1 or 0.
3) There is also an inhibitory input which, if is set to 1, will prohibit the neuron from firing
4) The neuron has a threshold value. If the sum total of all the inputs is larger than that of the threshold, then the neuron will fire otherwise not.

The MCP model had very hard rules which reduced its flexibility in order to adapt and learn according to the data provided. Rosenblatt came up with modifications to the existing MCP neuron model in a supervised sense that actually allowed the MCP model to figure out the appropriate weights for itself based on the data it is being trained on. The ability of the modified MCP model was the ability to perform classification in binary form. Rosenblatt's model was the one that was able to distinguish between two classes only if they were linearly separable.

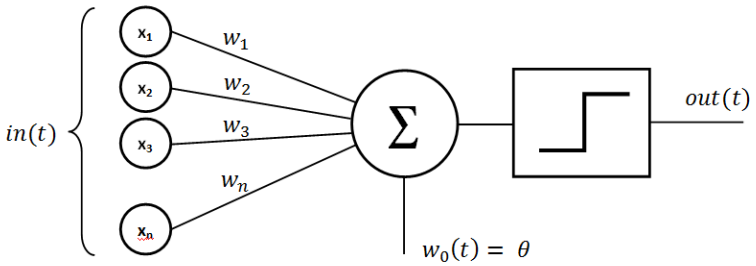

Fig. 3.2 A basic perceptron model [38]

Following the works of Rosenblatt, Widrow and Hoff in 1959 came up with work of a feedback learning algorithm known as the LMS algorithm [40] in which they suggested a methodology of updating the values of the input, based on the output of the system. Rosenblatt's single-layered perceptron along with the LMS algorithm led to the creation of the multi-layered perceptron by Minsky & Papert [41]. They showed that if a single-layered perceptron is able to classify data into two classes based on the formulation of a linear function, then a multilayered application of the same single layers perceptron will be able to grab information and classify multiple classes based on the multinomial function due to those multiple layers.

Input: Training examples $\{\mathbf{x}_i, y_i\}_{i=1}^{m}$.

Initialize $\mathbf{w}$ and b randomly.

while *not converged* **do**

    # # # Loop through the examples.

    for $j = 1, m$ **do**

        # # # Compare the true label and the prediction.

        $error = y_j$ - $\sigma(\mathbf{w}^T \mathbf{x}_j + b)$

        ### If the model wrongly predicts the class, we update the weights and bias.

        if *error != 0* **then**

            ### Update the weights.

            $\mathbf{w} = \mathbf{w} + error \times x_j$

            ### Update the bias.

            $b = b + error$

    Test for convergence

Output: Set of weights $\mathbf{w}$ and bias b for the perceptron.

Fig. 3.3 A perceptron learning algorithm [41]

3.3 BACKPROPAGATION

Works in the field of backpropagation started way back in the 1969s when P.J Werbos [42] in his doctoral dissertation, explained a new methodology of reducing errors using the concepts of partial derivatives and updating the error equation. Following immediately Seppo Linnainmaa in his work showed, what we today know as the modern form of backpropagation [43]. In his work, he described, both in the form of an algorithm as well as in analytical form on how to determine the Taylor series expansion of an accumulated rounding error with respect to the local rounding error [43].

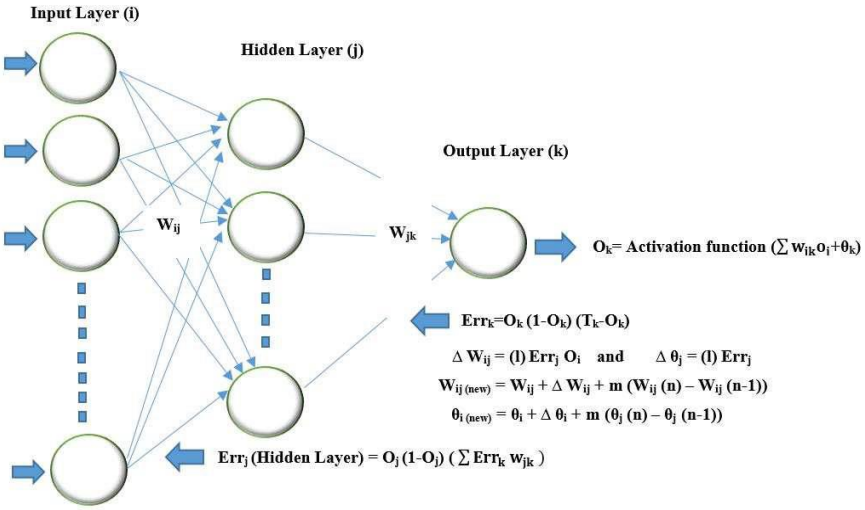

Fig. 3.4 Backpropagation algorithm for a neural network [43]

The way in which backpropagation works is that for a given input and output it aims to approximate a non-linear relationship between them by updating the weights of the network internally. When breaking down the backpropagation algorithm, two methods take place in the process. One of it being, the feedforward operation, which is simply put, the input data is provided to the network and we get a corresponding output. There is no weight adjustment happening at this stage of the process. After the feedforward operation takes place, the input that was originally given to the network is compared to the expected output. Every node in some shape or form is added to the error at the output, that is when the output is sent backward which travels layer by layer, node by node to the input. Once all the errors are determined, the magnitude of the error is then used as a parameter to update the corresponding weight at each and every node of the network. The way in which the network updates its weights is by trying to minimize its error by minimizing, what we call an error function. These minimization techniques are generally gradient descent approaches.

3.4 NEOCOGNITRON

The first-ever creation of a convolutional neural network can be accredited to Kunihiko Fukushima in 1979 [44]. In his work, for the pattern recognition in the visual domain. He proposed a network that he termed “learning without a teacher” [44]. The neural network proposed by Fukushima [44] tried to learn the similarities in the form of geometries. He gave this network the name of ‘neocognitron’. It was argued in the work that after the network is able to arrange itself, it was in a form of how the nervous system

is for the human in the context of visual perception. The structure of the model was very similar to what we see in conventional neural networks consisting of an input layer and then interconnected hidden layers which learn the patterns and weight updates. The model also showed the concept of plasticity as denoted by Hebb as the ability of neurons to change when the process of learning takes place.

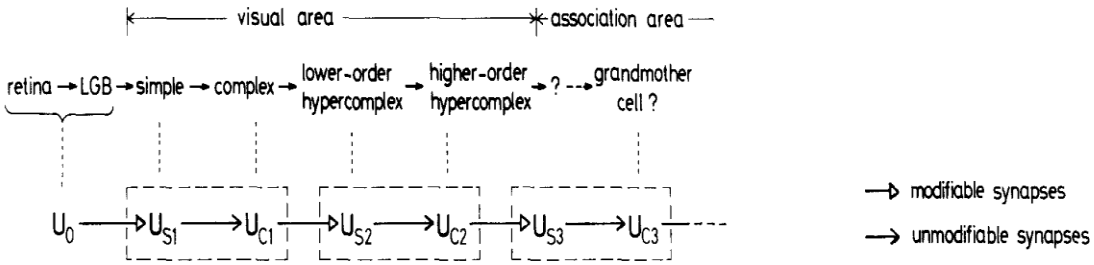

Fig. 3.5 Figure denoting the relationship between the hierarchical model and neocognitron [44]

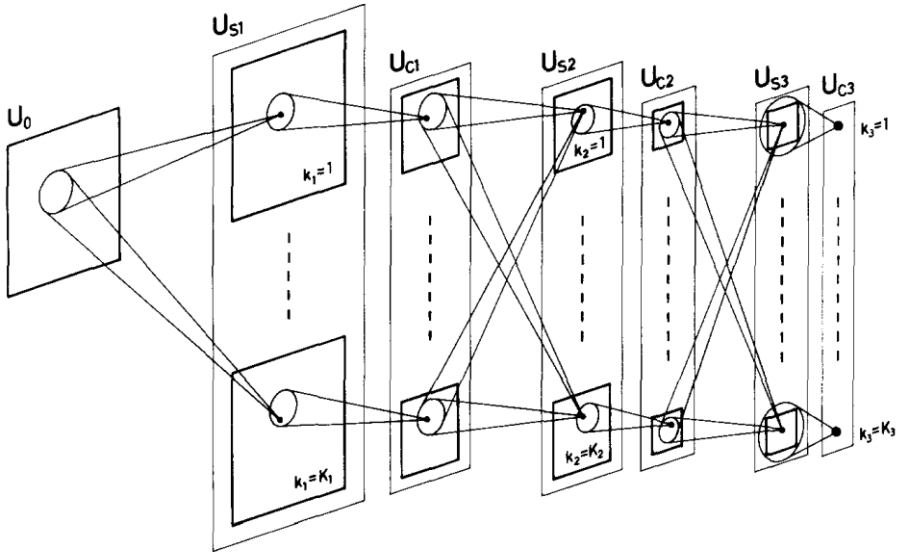

Fig 3.6 Diagram outlining the interconnections layerwise in neocognitron [44]

3.5. LeNet: THE FIRST MAINSTREAM CNN

After the works of Rumelhart [9] in the field of backpropagation, LeNet-5, the name was captured from that of the creator, Yann LeCun [45]. Before the creation of LeNet-5, the author, in one of his earlier works in the same year actually [46] performed experiments on a varying number of layers in the neural network architecture [46] with the use of back-propagation as the main learning algorithm for the network, which was showcased by Rumelhart in the year 1986. Backpropagation was developed long before it became the staple learning algorithm of neural networks but mainly had its applications in different fields of work. The main idea behind the concept of backpropagation was always in the field of optimality in a control system.

For the work done by LeCun [45], they applied a backpropagation algorithm to use in a real-world scenario and that was to recognize digits from handwritten letters that were collected from the US Department of Mail. The neural architecture instead of being fed with a vector of fixed length was actually fed directly with the image of digits, which helped showcase that backpropagation has the ability to handle large collections of low-level data.

The database of the images was made up of 9298 images that were extracted from the handwritten codes that appeared on the US mails. The training dataset was comprised of 7191 images and the rest 2007 images were used for testing. The best thing about this dataset was that it was full of ambiguous and non-classified images as well [45]. For the preprocessing step, the images that initially were gathered from the zip codes had a variable dimension of 40 X 59. They applied a transformation that was linear in nature, to convert the variable size input image into 16 X 16 images. The images resulted in multiple values of different gray levels, thus they were also converted into the range of 1 to -1.

The network was a simple multilayered neural network that used backpropagation as the learning algorithm and the whole network and every layer was trained using backpropagation. The input layer of the network took 16 X 16 images as input and the output consisted of 10 output nodes because of 10 digits in the number system. The network was forced to extract small (local) features which when aggregated together got converted into large features. Different features were present in different locations in different images. Following this the feature detectors were set to extract any instance of the digit in the images irrespective of the location of the instance in the image because for the classification case it was not necessary to pinpoint the exact location of the instance in the image.

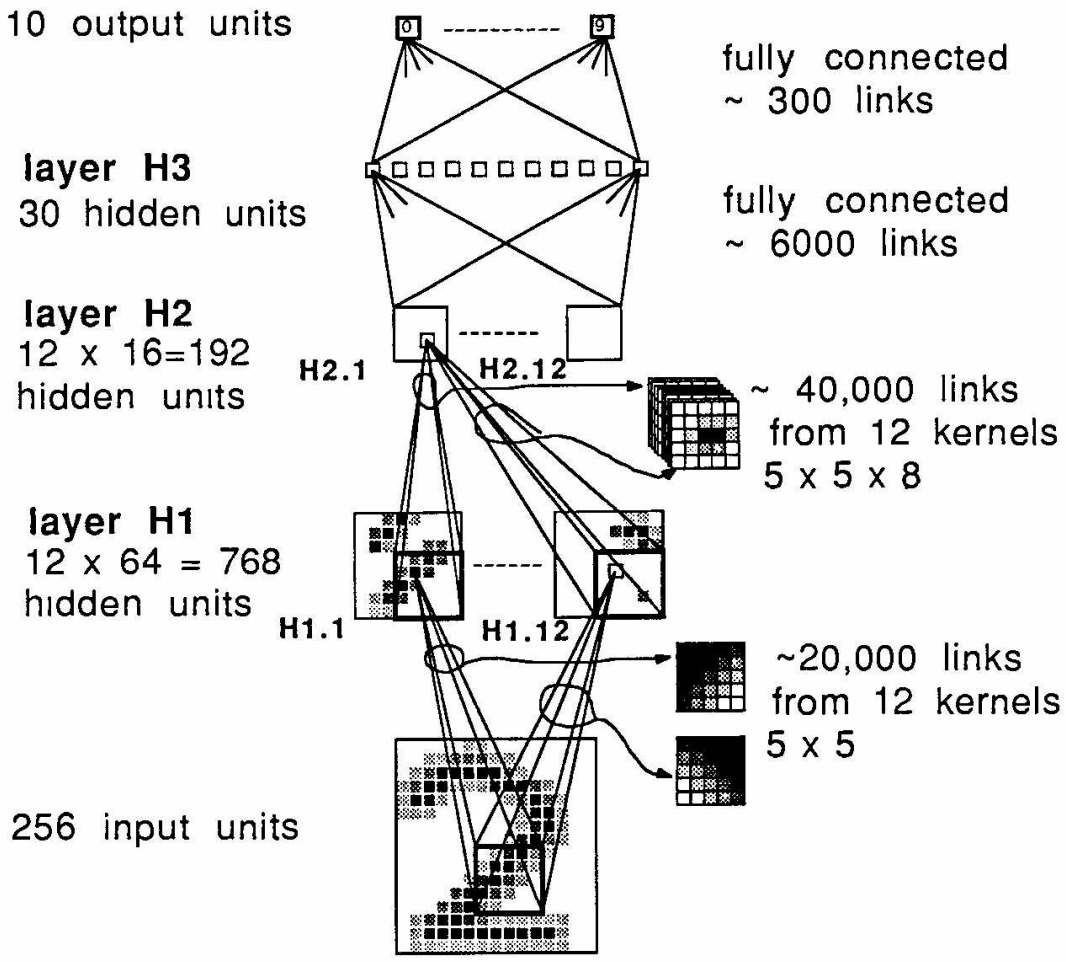

Fig. 3.7 LeNet Architecture [45]

3.6. AlexNet AND GoogleNet

After the work done in LeNet [45], the research in the field of CNNs was not progressing as some might have hoped for. Still after the showcase of the ability of CNNs to generalize and extract features, people were still using handcrafted feature extraction methods to classify objects. The biggest roadblock that most of the researchers especially in the academic community faced was the non-availability of a large-scale collection of images for training their models. So one of the breakthroughs that pretty much has become the SOTA dataset for any new image recognition is ImageNet Large Scale Visual Recognition Challenge [47]. The challenge started in 2010, by a team of researchers led by Dr. Fei-Fei Li, was the first of its kind competition in the field of image classification

and much more for the all the researchers all around the world to come up with CNN models to test them on this large dataset.

One of the very first architectures to reach the state-of-the-art error rate in the ILSVRC [47] was in 2012 by the name AlexNet after the creator Alex Krizhevsky in the paper titled “ImageNet Classification with Deep Convolutional Neural Network” [47]. This one of the very first architectures that successfully used a non-saturating nonlinearity in its architecture. The AlexNet [47] used ReLU [48] architecture which was showcased in the works by Nair and Hinton [48] in 2010. AlexNet [47] was also one of the first architectures to fully utilize the parallel processing ability of GPU.

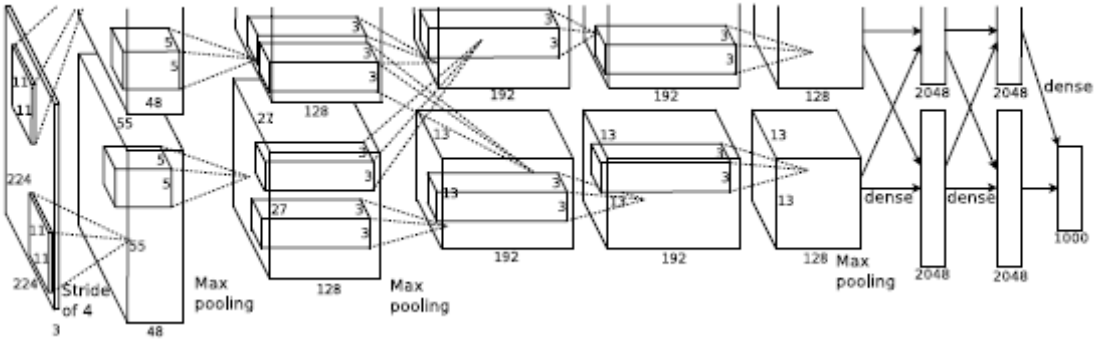

Fig. 3.8 Architecture of AlexNet [47]

AlexNet [48] was the first architecture that showed the various aspects of a convolution network that can be exploited to make the neural network learn faster and give out better results with a top 5% error rate of 15.3%. Following this work subsequent architectures came up in the next years in the ILSVRC [47] competitions like ZFNet [50] posted even a better error rate of 11.2% in the year 2013. Similarly, in the year 2014, two of the architectures came out on top. One of them being, GoogLeNet [51], otherwise also known as the architecture to give rise to a block with multiple connections known as an “Inception” block to make the network deeper which gave a top 5% error rate of 6.66% [51]. The other being VGG [52] created by the Visual Geometry Group of the University of Oxford which gave a top 5% error rate of 7.3%. [52] Both the architectures were the first ones to drop down the top 5% error rate to single digits

Going a little bit in-depth about the VGG [52] architecture. This architecture though wasn’t the winner of ILSVRC [47] competition but went forward to become one of the most famous convolutional neural network architectures with the paper being cited more than 77 thousand making it one of the most cited papers ever. The main reason for

the popularity of VGG [52] architecture was its simplicity. Though GoogLeNet [51] won the competition, the concept of multiple connections with multiple convolution kernels was still new. Below are the building blocks of GoogLeNet [51].

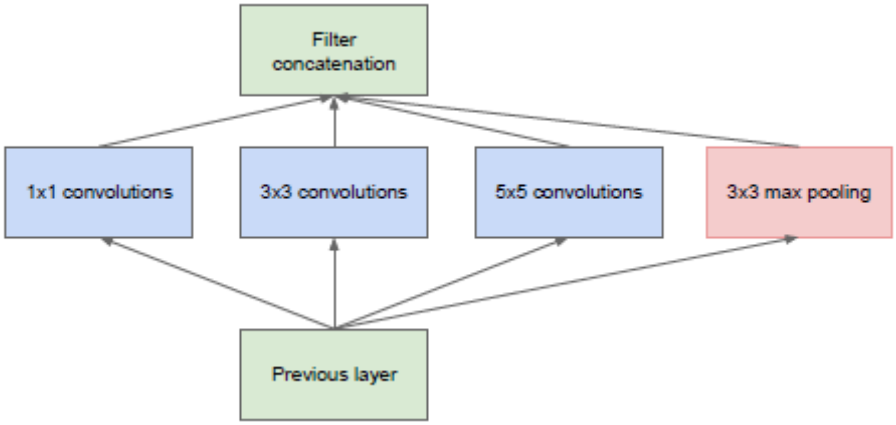

Fig. 3.9 Inception Block [51]

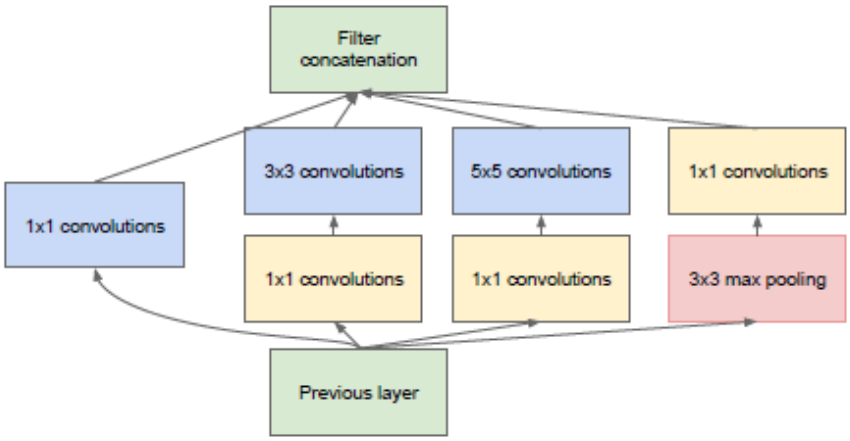

Fig. 3.10 Inception Block with dimension reduction [51]

3.7. VGG

VGG [52] in itself is a very simple sequential architecture with has one to one connection will all its previous layers. The convolution architecture of VGG [52] took a fixed input image size of the dimension 224 X 224. In the preprocessing step while training the network architecture, the authors deducted the RGBs mean value from each input pixel. All the previous architecture used big kernel sizes of 11X1 in case of AlexNet

[48] and 5X5 in the case of GoogLeNet [51]. But VGG [52] used the smallest sized filter of dimension 3X3 because this is the smallest size of filter kernel in the stacks of convolution layer with the ability to move in all 4 directions to generate feature maps based on the input image being fed into the network. The stride of the kernel in the convolution layers is fixed to 1 and the padding used in the convolution layer is fixed to a one in the sense that the spatial resolution of the image is preserved once the convolution operation happens on the image and feature maps from previous layers [52]. Regarding the pooling operations, 5 max-pooling layers [52] are connected after the batches of convolutional layers. The penultimate layer before the classification layers is a max-pooling layer with a window size of 2X2 and the stride of the window set at 2.

The output feature map size before the classification layers comes out to be 7 X 7 X 502. Following the feature maps, two fully connected layers with 4096 nodes are with ReLU and after that a 1000 node fully connected layer for the ImageNet dataset because it contains 1000 classes of images. The activation used in the last layer is sigmoid.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig. 3.11 The Layer structure of various VGG types [52]

3.8. ResNet

Following the creation of VGG [52] in 2014, researchers all around the world started building neural networks deeper and deeper by stacking as many layers in the network as possible. But one of the breakthrough work by Kaiming He [53] in 2015 turned out to be the winner of the ILSVRC [47]. A similar work having the concept of skip connection was proposed by Shrivasta [54] in the paper titled “Highway Networks” [54]. Similar to this the idea of using skip connections [53] in the interconnection block provides an alternate path for connections.

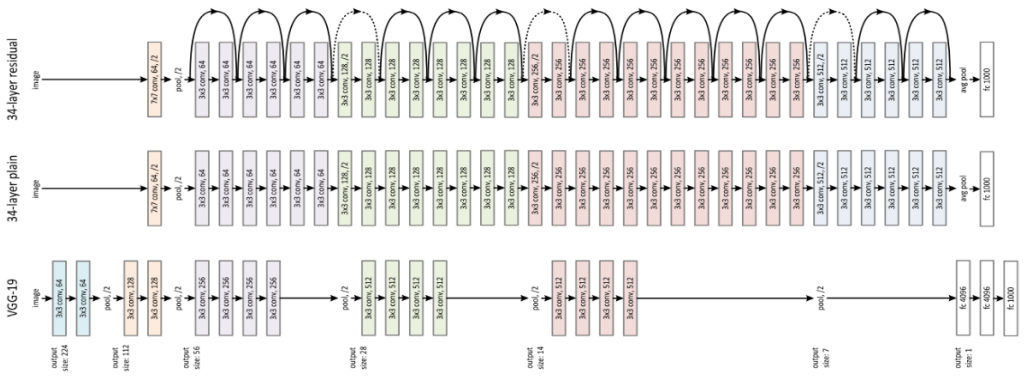

Fig. 3.12 34 Layered ResNet Architecture [53]

The idea of ResNet [53] rose to prominence because of two main reasons:

1) When the number of layers in a neural network is kept on increasing, the error rate decreases as the accuracy increases and then becomes stagnant. As any more layers are added, the accuracy instead of increasing actually starts to degrade rapidly and this as a shock is not due to the overfitting of the model
2) The second reason is the notorious problem of vanishing gradient. As the number of layers in an architecture keeps on increasing, the gradients calculated as a reason for backpropagation might become so small before reaching the earlier layers, that there is no significant inference to be drawn from that with no weight updates.

So to solve the above two issues, researchers from Microsoft suggested ResNet [53]. Since we can model neural networks as one that approximates functions, the concept of ResNet [53] was that instead of learning the function of the specific input to a

very deep architecture, instead of learning the true function, we instead try to learn what we call the residue [53].

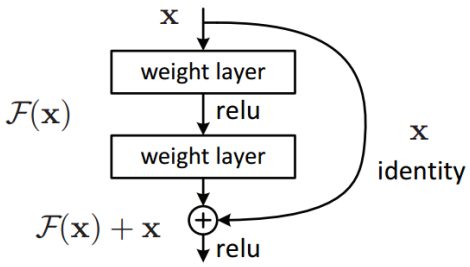

Fig. 3.13 A residual block from the ResNet Architecture [53]

From the above figure, we can see that we aim to learn the function *F(x)* in traditional methods. But in the residual blocks, we aim to learn *H(x). H(x)* being the true distribution and *x* being the identity connection. Though the residual block overall aims at learning *H(x)*, looking closely at the image above, there is an identity connection joining the input to the output, and the layers are actually learning *F(x)*, because we are trying to minimize the error between the input and output and that can be easily achieved by the Eqn. 3.1

$$H(x) = F(x) + x \tag{3.1}$$

As we see from the figure and from the above equation, the skip connections actually allow the gradients with large values to propagate easily to the first few layers, which in the case of a traditional sequential connection falls through due to vanishing gradient. This also helps the layers to learn faster. From the experiments conducted by the authors, they came up with multiple approaches for the residual block as given in the figure below. The best result was showcased by residual blocks with full pre-activation.

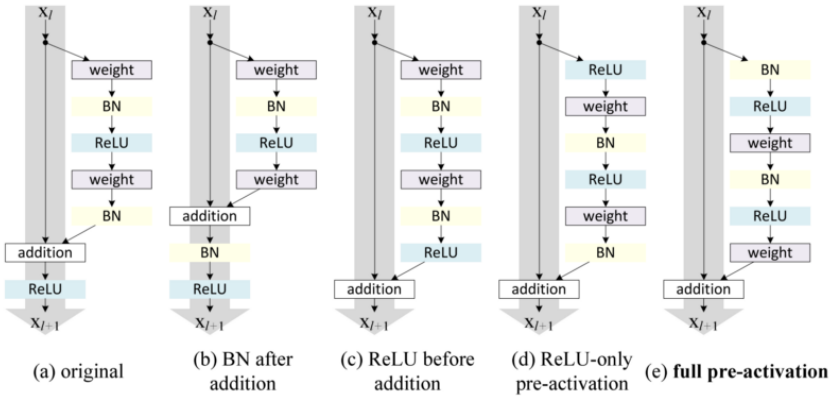

Fig. 3.14 Different varieties of residual blocks [55]

3.9. OBJECT DETECTION

Some of the earlier work in the field of object detection can be found in the field of single-stage detectors like the works of Viola and Jones [56] in the year 2001 in IEEE's CVPR 2001. In order to honor the two researchers, the detector was called the Viola-Jones Detector [57]. The approach of this detector was very simple and easy. Since this was mainly focused to find the faces of humans, the detector was a simple window sliding over the whole image. Passing through every location possible on the image to find the human faces. The detector was way more powerful as compared to the hardware available for performing such tasks. Talking about the technical part of the detector, the VJ [57] detector used Haar wavelets to represent features of the images and make the size of the window independent of the computational complexity. The detector also used an algorithm to select the features from the image and used cascades [57] to reduce the overhead of computation.

The work in the field of traditional single-stage detectors continued with the use of HOG detectors proposed in the year 2005 [58]. HOG stands for Histogram of Gradients [58]. One of the areas that the HOG improved was the area of invariance of features and the non-linear nature of the objects present in the image. The HOG [58] uses the concept of gradients in other words slopes. The different intensities of the pixels are because of the groups of tightly packed pixels providing us the information about the location of the objects in the image. The detector uses the same sized window to find out multiple objects in the image but performs a rescaling operation on the input image. Next in line was an object detection model known as DPM [59] which was an improved version of the HOG detector. The main idea of detection that DPM [59] followed was "divide and conquer" [59]. The training of the model can be simple breaking down the image into various objects and when inferencing the model can be consolidating the decomposed objects from the images. R. Girshick improved the basic DPM to detect multiple objects more inclined toward the real-world scenario of images.

All the detection methods discussed above tend to follow the pattern of using hand-crafted features to detect objects in the images be in sliding the window, using gradients of the objects, be it dividing the image, and then recombining it again to detect the objects in the images. The biggest drawback in the approaches using hand-crafted feature extraction methods is the computational complexity and the amount of time the computation system required to detect the objects in the images. After the year 2008, when DPM was formulated the area of object detection plateaued for quite a few years. The main reasons being roadblocks in the form of creating handcrafted features and a huge

amount of computational resources required to locate the objects. But as soon as AlexNet [48] in the year 2012 showcased the prowess of neural networks and especially convolutional neural network, it open up the floodgates for the researchers all around the world to get down and get their hands dirty in order to use the absolute power of CNN for object detection.

One of the works that started the use of CNN in literature can be attributed to Overfeat [60] which was proposed in the year 2013 by the team of Yann LeCun. The work of Overfeat was overshadowed by one of the works of the authors who improved the existing approach of DPM, R. Girshick. In the year 2014, R. Girschick, and his team proposed a two-stage approach for detecting objects by the use of CNN. They proposed regions in which the objects are supposed to be located and used a collection of granular information extracted from the extractor to predict the object's location. The first of his works was titled R-CNN [61] where the "R" stands for regions. The formulation of R-CNN [61] was very simple. Initially use a search algorithm to propose as many regions as you can in the image, in this case, the authors used a selective search algorithm [62]. After this, the proposal regions generated by the search algorithm are rescaled. After the regions are rescaled, the images are then fed into feature extraction networks or CNNs previously trained on ImageNet [47] dataset for the purpose of extracting features. After the features are extracted from the images, the pixel data were required to be classified. The pixel data to be predicted were fed into SVM classifiers, and calculated whether an object is present inside the proposed region or not and whether the object lies in one of the object categories or not. The R-CNN [61] was used on VOC2007 dataset with the detection metric of mAP(mean Average Precision) jumping from 33.7% in the case of DPM to 57.5% in the case of R-CNN. The biggest drawback of this sort of approach was the creation of a very large number of region proposals which slowed down the detection process and in turn also increased the computational requirement for the network.

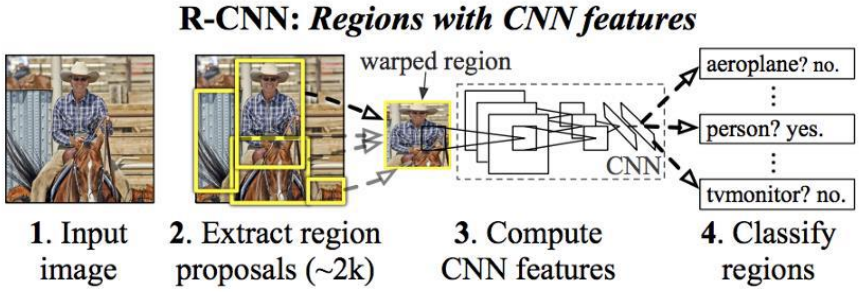

Fig. 3.15 R-CNN Object Detection Method [61]

To overcome the huge number of proposals, in the year 2014, Kaiming He and his team proposed SPPNet [63], an acronym for Spatial Pyramid Pooling Networks [63]. Earlier detection methods forced the researchers to fix the input size of images when giving input. The fixed-length input was necessary to create a fixed-length output. But SPPNet [63] proposed pyramid pooling layers using CNNs to generate representations of the input images to be of fixed length irrespective of what dimension the input images are thus saving computational cost in rescaling the images. The improvement over RCNN [61] was that the feature maps generated over the entire images were calculated only once from which the detector were trained on the representation saving computations in the process of doing so. The biggest drawback of this technique was that the architecture only updates the weights of the final fully-connected layers while not considering and of the previous layers.

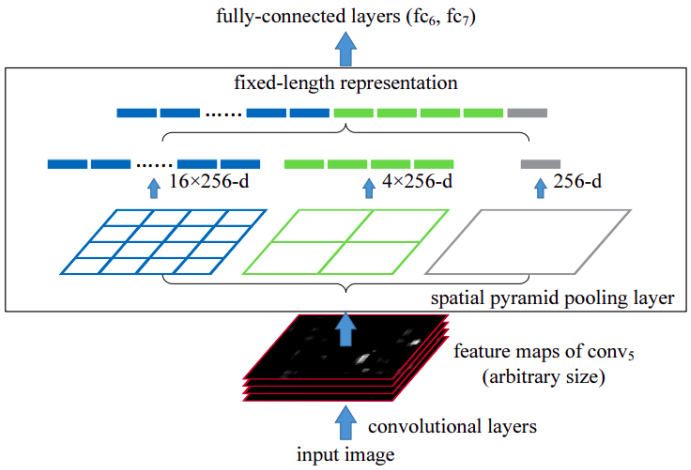

Fig. 3.16 Spatial Pyramid Pooling Layer [63]

In the next work improving on the disadvantages of R-CNN [61] and SPPNet [63] was Fast R-CNN [64]. The Fast R-CNN introduced to the community of what we today know as Region Proposal Network (RPN) [64] which contributes computationally nothing to the overhead like other approaches discussed earlier. The Fast R-CNN combined the multiple blocks of other approaches into one by the use of RPN. The main advantage it had was higher detections than previous approaches [64], used a single-stage training methodology [64], updated the weights of all the previous layers [64], saved space on the disk by bypassing caching [64].

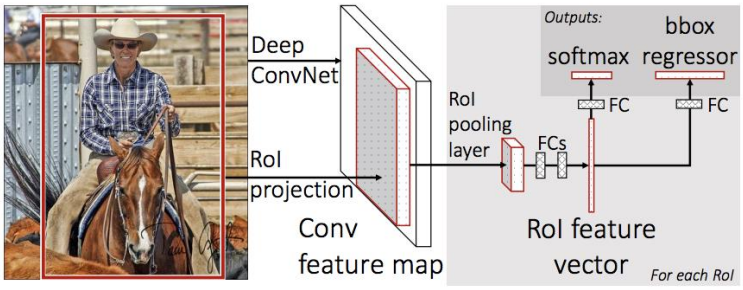

Fig. 3.17 Fast R-CNN Object Detection [64]

3.10. FASTER R-CNN

From the above two approaches from the family of using region proposals, both R-CNN [61] and Fast R-CNN [64] use selective search approaches to find out region proposals which are huge in number. It is a very slow and time-consuming process which in turn affects the network performance. Thus in the year 2015 Shaoqing Ren, along with R. Girshick, Kaiming He, and Jian Sun introduces “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks” [65] at that year’s NeurIPS Conference.

Working on the deficiencies from the previous works, the region proposal networks tried to reduce the computation bottleneck caused due to previous region proposals. The way the algorithm works is first it takes an input image with no specific input shape and then gives an output in the form of bounding boxes as the proposals where the object might be present. The aim as mentioned for the RPN networks is to have shared computations, they used VGG [52] because it has 13 convolutional layers with shareable weights and parameters. The region in the context of this specific work takes into consideration just rectangular boxes.

So the region proposal works in the sense that, the convolutional feature map output of the last layer of the feature extraction network is get into a small network to generate the region proposals [65]. The small network used consists of fully connected layers that take in input from a connection with $n \times n$ window of convolution feature map. The network sliding over the feature map tries to create a one-to-one map into a lower dimension which is 502 for VGG [52]. This lower-dimensional vector is then given as an input into two more fully-connected layers [65] one of them being a bounding box

regression layer and a bounding box classification layer. ReLU activation is used as the output of the above-mentioned $n \times n$ convolution layer [65].

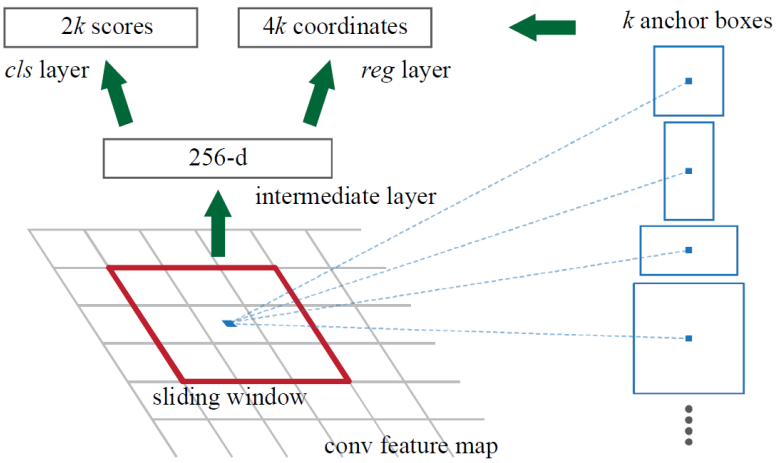

Fig. 3.18 A Region Proposal Network Architecture in Faster R-CNN [65]

When the window is slid over the locations, the detector tries to predict a *k* region proposals, the regression layer then encodes *k* box coordinates into *4k* number of outputs. At the same time, the classification layer gives *2k* scores as the output predicting the probability of the presence of an object or not. The proposals generated by the sliding window are turned into parameters which are then compared to *k* boxes that act as references and are thus known as *anchor boxes*. This approach uses 3 different scales and 3 different aspect ratios [65]. The feature map sizes of dimension $W \times H$ lead to a total of $W H k$ number of anchor boxes.

The implementation also samples 255 anchors from an image and uses that to calculate the loss function of the mini-batches used [65] with the positive and negative anchors set at a ratio to go *up to* 1:1 [65]. The cut-off is set at 128 samples being positive. If the number is lesser than this, then the positive and negative images are padded together in the mini-batches. The detector implementation starts with random initialization of all the new layers being drawn from a Gaussian distribution with the mean zero and 0.01 standard deviation [65]. The shared layers are loaded with pre-trained ImageNet weights. The learning rates used for 59k mini-batches were set at 0.001 and for the nest 20 k mini-batches at 0.0001 [65].

The detection approach used is actually Fast R-CNN [64]. RPN [65] and Fast R-CNN [64] both of them, independently trained with the objective that the convolution layers present in their architecture with change their weights in contrasting ways. Thus a methodology was required, that should get shared between these two networks, such that the networks instead of being trained independently on their own, would get trained and the weight updates being shared between them. There was a very high chance that when training both the networks by combining them into a single network the mechanism that proposes the regions might not converge at all. So they developed these 4 steps [65]:

1) First, they trained the RPN using ImageNet weights and then fine-tuned the model end-to-end to propose the regions.
2) Second, they trained the ImageNet initialized Fast R-CNN [64], which is the detection network in this case, using the regions proposed in the above step.
3) Third, they used Fast R-CNN [64] networks to initialize RPN, and then freeze the convolution layers that are shared among both of them and then tune finely the fully-connected RPN layers that are unique. Now it is clear that both the networks share the layers.
4) Fourth, they freeze the shared convolution layers, and finely tune the fully connected layers of the Fast R-CNN [64]. Thus forming a cohesive network.

All the experiments performed for training and testing was done on one of the standard dataset known as PASCAL Visual Object Challenge or in short PASCAL VOC [67]. The dataset used were from the year 2007 and 2012, with the dataset in 2007 containing 20 image classes, with 9,962 total images containing approximately 24,630 annotated objects and 11,520 images along with 27,449 annotated objects in 2012 [67].

CHAPTER 4

REINFORCEMENT LEARNING

4.1. INTRODUCTION

When it comes to the main idea behind the use of machine learning, one of the core areas that takes precedence is the idea of decisions and that too if it aims at following some sort of sequence. Before the creation of reinforcement learning as a separate branch of machine learning, the works were used to be published in psychological magazines the reason being reinforcement learning in its core is a mixture of many areas including psychology. One of the earliest works can be quoted to Thorndike [68] on how the animals associated processes with each other based on what sort of environment they stayed in and the approach they took based on the environment around them. In one of the works again by Thorndike [69], he clearly showcased that animals were more likely to do the works of what they were rewarded better as supposed to what they were not rewarded [69]. This can be thought of as one of the major driving processes when we talk about reinforcement learning. Be it a small kid, bet it a 90-year-old grandpa, all of the work one does during a lifetime is a cause and effect relationship of performing some task in lieu of some reward. Now it's not necessary that whatever actions one performs are going to be rewarded in a positive sense. Let's take an example where a small kid. If he falls while playing and gets hurt, he gets s negative reinforcement that forces him to be cautious in the future. Similarly, when he achieves something and gets rewarded, he learns that as a model of positive reinforcement. So this approach is where the individual learns as a consequence of taking a specific action. This was one of the approaches in reinforcement learning.

Another approach that gave an insight into the early stages of reinforcement learning is the area of optimal control. As in the previous approaches, we can see that the individual is learning but in this sort of approach, there is no sense of learning. Only one

tries to find the best strategy for an optimal strategy to perform things in a quick and easy sense. Thus when talking about optimizing the process, one of the equations we talk about is the Bellman equation designed by Richard Bellman. The formulation of the equation follows a dynamic system and its state and a function giving a value of it [70], the returning function-based equation known as the Bellman equation [70]. The type of methods that were used for solving these control optimization problems were known as dynamic programming because the optimization happened on dynamic systems which Bellman mentioned in his work [70]. When such problems of optimal controls were instead of being continuous, made discrete, the problem was given a new name, now known as Markov Decision Process or MDPs [71]. Continuing this work of reinforcement learning, Ron Howard in the year 1959 created an iteration technique of policies for the Markov Decision Process [72].

Continuing on the third approach that leads to the creation of modern-day reinforcement learning is the concept of temporal-difference learning [70]. The idea behind this sort of learning method is to use the difference between the successive estimation of the same quantity with the difference between them being in terms of time and nothing else. The origins again can be traced back to the ideas from the area of animal psychology using the concept of *primary* and *secondary* reinforcer [70]. The first application of this psychological approach in practice was by Arthur Samuel [74] in the year 1958 in his paper titled, "Some studies in machine learning using the game of checkers" [74]. In order to prove his previous work [76], Minsky [75] extended the work by Samuel [74] to give his suggestion to connect artificial as well as the natural idea of secondary reinforcement theory [70].

Finally, all the work done in the area of temporal difference and optimal control was consolidated in the year 1989 by Chris Watkins with the introduction of what we today know as Q-Learning in his Ph.D. thesis with the title "Learning from delayed rewards" [77]. This brought together the trial-and-error approach of the first idea from the learning phase, the second idea of just control optimization, and the third idea of the temporal difference. In the year 1992, Watkins [77] along with Peter Dayan [78] officially introduced the work done by Watkins [77] to the world in the paper titled "Q-Learning" [78].

4.2. BASIC REINFORCEMENT LEARNING PROBLEM

When coming to terminologies regarding laying down the mathematical framework for reinforcement learning there are quite a lot of things to keep up with. An everyday reinforcement learning problem can be described as a discrete control process

with respect to time and which is stochastic in nature, where the agent is going to depend on the environment in the following steps:

1) Firstly the agent is present in a particular state in a given environment, $s_0 \epsilon S$.
2) It then gets information on an observation initially, $\omega_0 \epsilon \Omega$.
3) After the above two steps, at every time instance t, the agent then performs an action $a_t \epsilon A$.

From the above three steps, we see that we can relate to the different steps a reinforcement learning algorithm takes as same as the ones we take in real life. At the moment we are currently in a situation, we observe the environment we are in. then based on every observation in that environment, we take actions at specific time instances. Now based on the above three steps, in this case:

1) The agent with yielding a reward, $r_t \epsilon R$.
2) The agent move on from that initial state, $s_{t+t} \epsilon S$.
3) The agent gets a new observation, $\omega_{t+1} \epsilon \Omega$.

The above method of signifying a control problem was defined by Bellman in [71]. The above work was then comprehensively covered by Sutton and Barto in [70].

4.3. MARKOV DECISION PROCESS

Starting with the definition of a Markov process one can say, "If for a sequential state, the probability that the process will move into the next state depends just and only just on the current state of the process is currently and without taking into consideration of all the previous states the process have been through, then the sequential process is known as a Markov process". All the states are denoted with the letter S_t, where S denotes state and t denotes the instance of observation of the process.

$$P\,[S_{t+1}|\,S_t] = P\,[S_{t+1}|\,S_1, S_2, \ldots\ldots, S_t] \tag{4.1}$$

We generally deal with time-independent Markov processes in reinforcement learning, thus the transition probability [70] is considered independent of time.

$$P\,[S_{t+1} = s'|\,S_t = s] = P\,[S_t = s'|\,S_{t-1} = s] \tag{4.2}$$

The Markov chain is considered as a tuple of states and corresponding probabilities. Here the *S* denotes the finite number of possible states and *P* denotes the possible transition probability of the agent going from one state to the other. Some of the terminologies used in a typical reinforcement learning system are:

1) Agent [70]: The entity that actually is existing in an environment, from which it learns, performs actions and gets rewards based on the action it takes.
2) Environment [70]: The surrounding provided to the agent in which it stays and learns from it. An example can be an agent used in a computer vision task that has the image as its environment.
3) State [70]: The current position of the above-mentioned agent in any environment based on the current observation it takes into consideration.
4) Action [70]: The steps that the agent takes in an environment. In practical applications, the actions taken by an agent are generally discrete in nature. Like a robot, it can be moving forward, backward, left, right, etc.
5) Reward [70]: Every action an agent takes in an environment, in a particular state, is to get the maximum reward which it aims to do either implicitly or explicitly.

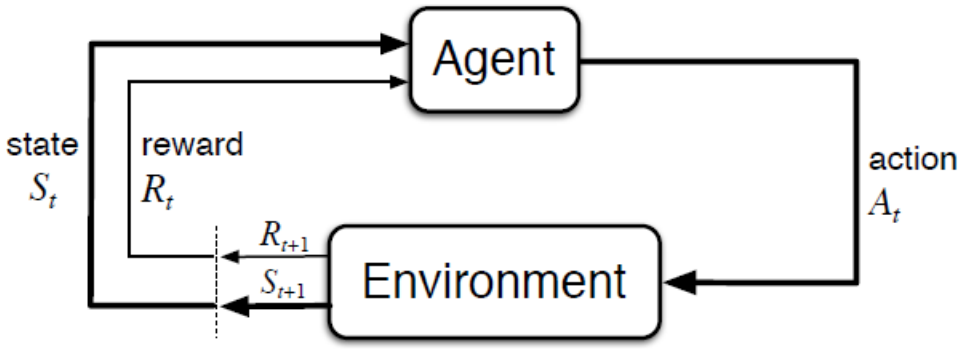

Fig. 4.1 Agent-Environment Interaction in a Markov Decision Process [70]

A Markov Decision Process is a collection of 5 arguments (*S, A, T, R,* γ) [79].The *S* collection of states, *A* is the collection of action, *T* is the collection of transition probabilities in between various states, *R* is the function defining rewards, and γ is the discount factor.

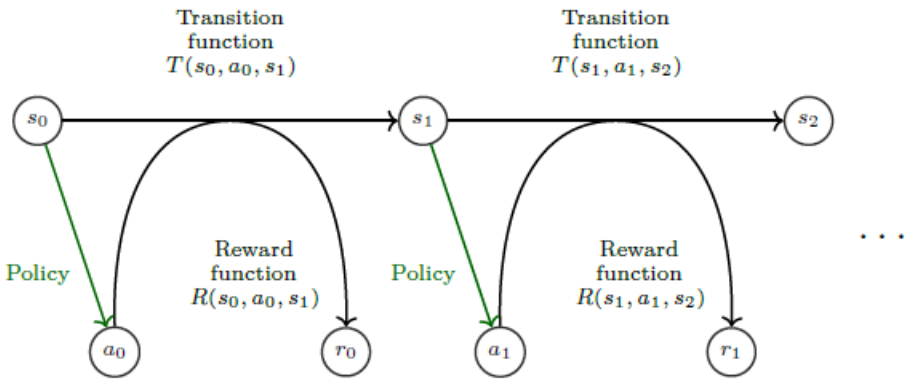

Fig. 4.2 A working flow of an MDP in action [79]

Now how will the agent know which action to take when in a current state in an environment? There has to be a guiding method of doing so. Such guiding methods are known as *policies*. We define a policy in a reinforcement learning setting as a definition of how an agent will select an action. Policy in an RL is bifurcated into the criteria of being either stationary or non-stationary.

Stationary policies as the name suggests are stationary. Here the word stationary is mentioned in the sense that the dynamics of the environment in which the policy that is being followed do not change with time. It also suggests that when taking any sort of decision, whenever specific conditions are met, the agent always will take the same decision independent of time. It can be thought that even though the agent takes multiple decisions, the probability corresponding to those decisions remains the same. The use of non-stationary policies is time-dependent and is very useful when the total reward the agent will be able to farm is limited to a specific number of steps in it. So to summarize, stationary policies use an infinite time horizon for cumulative rewards non-stationary policies take into consideration a finite time horizons for cumulative rewards.

One of the other ways we can distinguish policies is by separating them into either deterministic or stochastic [79]. The policy in every place is denoted by π. So, talking about the deterministic policy it is a mapping of just from a state to action in that particular state and is given by $\pi(s)$: $S \rightarrow A$. When it comes to stochastic policy, it is bound to use probability and is defined by $\pi(s, a)$: $S \times A \rightarrow [0,1]$, where the policy definition $\pi(s, a)$, defines the probability that when the agent is present in state s, it can take action a. Definition of policy is also given by:

$$\pi (a \mid s) = P [A_t = a \mid S_t = s] \tag{4.3}$$

When learning a policy of reinforcement learning, the agent includes one or more components listed below:

1) It might take into consideration the value function calculated, which gives an indication of the quality of every action the agent takes or the quality of every state/action pair.
2) It might take into consideration about the representation of the policy function given by $\pi(s)$ or $\pi(s, a)$.
3) It also might take into consideration regarding the model of the environment, together with a planning algorithm [79].

When the reinforcement learning agent takes into consideration the first two points under consideration, then that sort of reinforcement learning approach is called model-free reinforcement learning [79]. When the last component is chosen, then the reinforcement learning approach is known as model-based reinforcement learning [79]. The indirect approach uses an environment's model.

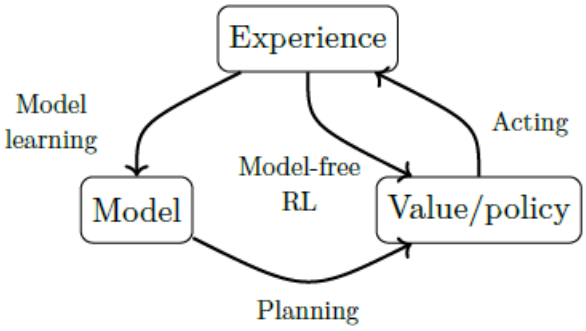

Fig. 4.3 Different reinforcement learning approaches [79]

4.4 Q-LEARNING

Q-learning as a methodology was introduced as Ph.D. thesis of Watkins in the year 1989 [77]. When first created the Q-learning algorithm created a lookup table containing every Q-value of the given function by the agent present in a given environment. The equation used to learn the Q-values is given by :

$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k} \mid s_t = s, a_t = a, \pi\right] \tag{4.4}$$

The Q-Vales collected above are generally stored in a form of lookup table. When trying to find out the optimal values of the Q-function, the Q-learning algorithm uses the Bellman's Equation for finding out the value of the Q-function given by the unique solution [79] *Q*(s, a):*

$$Q^*(s, a) = (BQ^*)(s, a) \tag{4.5}$$

Here the variable *B* signifies the Bellman operator that maps any function *F:* $S \times A \rightarrow \mathrm{R}$ into some other function $S \times A \rightarrow \mathrm{R}$ that is given by the equation below [79]:

$$(BQ)(s,a) = \sum_{s' \epsilon S} T(s,a,s')\big(R(s,a,s') + \gamma \, max_{a \epsilon A} Q(s',a')\big) \tag{4.6}$$

For the convergence of the above *Q** value there is a proof mentioned in the work by Watkins and Dayan in [78] on the condition given below as:

1) The representation of pairs of states and actions is discrete in nature.
2) Sample is taken out of all the actions that take place in all the states which makes sure there is ample exploration done by the agent in the environment.

Output: action value function Q
initialize Q arbitrarily, e.g., to 0 for all states, set action value for terminal states as 0
for *each episode* **do**
    initialize state s
    for *each step of episode, state* s *is not terminal* **do**
        $a \leftarrow$ action for s derived by Q, e.g., ϵ-greedy
        take action a, observe r, s'
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
        $s \leftarrow s'$
    end
end

Fig. 4.4 A Q-learning algorithm [81]

4.5. DEEP Q-NETWORKS

Now calculating and storing such a large number of Q-values will be just impossible, if at all the number of states that the agent can go is large and the number of steps the agent can is large, then the look-up q-table will be just infeasible both in terms

of storage as well as in the sense of computation. So in comes deep neural networks. As we all know neural networks are nothing but function approximators. They try to approximate a function that is best able to generalize and represent whatever is given as input in the network. Keeping in line with the various functions that the neural network tries to learn, using this in conjunction with the functions that one approximates in a typical reinforcement learning like value functions, finding out the optimal policy, or finding out the best state transition function or calculating the reward function such an approach of using deep learning and reinforcement learning together is what we today know as deep reinforcement learning. The parameters Θ in the case of reinforcement learning can be equated to the weights in the neural networks. Some of the approaches that have been used in this field covering the area of deep reinforcement learning start with an infamous paper in the year 2013, "Playing Atari with Deep Reinforcement Learning" [82]. Owing to the works of breakthroughs caused in the field of computer vison, the use of deep neural networks trained on large amount of datasets were required for efficient training. When raw inputs are directly fed into the networks and using stochastic gradient descent, these approaches yielded very good results. Based on the success of these approaches, the authors describe the use of interconnection between a deep neural network and a reinforcement learning algorithm working directly on an RGB image given as an input to the network and processing those updates based on stochastic gradient methods [82]. They directly feed into the network a pre-processed input of image into 110 × 83 and the RGB turned into grayscale with the final input dimension of the image fed into the network standing at 83 × 83. So when feeding into the Q-function [81], the pre-processing step discussed above is used and 4 frames from the history are then put together and given as input. For the architecture, for every expected action there is possible a standalone output unit is created. Thus for every input state given, the output gives a corresponding prediction of the Q-values [82]. So the convolutional neural network they trained on with the approach in [82] research work they coined the CNN as Deep Q-Networks (Deep Networks for neural networks and Q for Q-learning).

DQN in action was seen in one of the breakthrough papers in the year 2015 by the same group of researchers from Deepmind in the paper titled "Human-level control through deep reinforcement learning" [83].

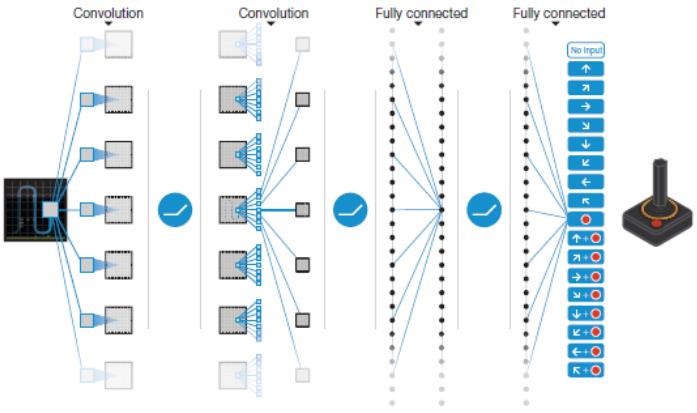

Fig. 4.5 A Deep Q-Network [83]

CHAPTER 5

PROBLEM STATEMENT

5.1. MOTIVATION

The work done in this dissertation is an extension of a future work suggested by the authors in the paper [84]. The work done by the authors is based on the premise discussed. When it comes to object detection, there are numerous approaches getting recognized each and every day in the research community. So when it comes to detecting objects implementing more and more advanced algorithms to find out objects from the given images is nothing new. But no approaches are being taken to correct the bounding boxes. This is what exactly the authors tried to do in their work in [84].

The authors used a reinforcement learning approach to correct incorrectly suggested bounding boxes based on experiments of incorrect human annotations, and loosely detected objects from any object detection algorithm. The main sector where this work can be used and employed in the area of logistics. When the logistic companies try to detect some objects or items they are using, there can be multiple reasons to use an object detection algorithm, maybe to keep an eye on the stocks, the damages being caused, and much more. Any object detection learning algorithm takes a very large amount of data to get trained which in turn takes quite a lot of time, money, and effort because humans have to be involved to actually annotate the objects by hand, and it's not possible for the businesses to update their detection algorithm every time a new one comes along. So what happens if the target object that the business aims to check on is changed to something different? In that case, they might have to completely redo the image dataset collection to include that new object in it. So to solve this issue this research work can be a good starting point. In the work done in [84], they used only non-overlapping images as their input and experimentation but in this work, I have tried to address the overlapping objects part to see that if objects are overlapping, is the reinforcement learning agent able to correct those bounding boxes.

5.2 INTERSECTION OVER UNION AND BOUNDING BOX

Intersection Over Union or IoU is a very important parameter when it comes to object detection of any sort. When the given original bounding box coordinates totally overlap with the coordinates of the detected objects the IoU value is said to be 1 and if the detection totally missed the original object it turns out to be zero.

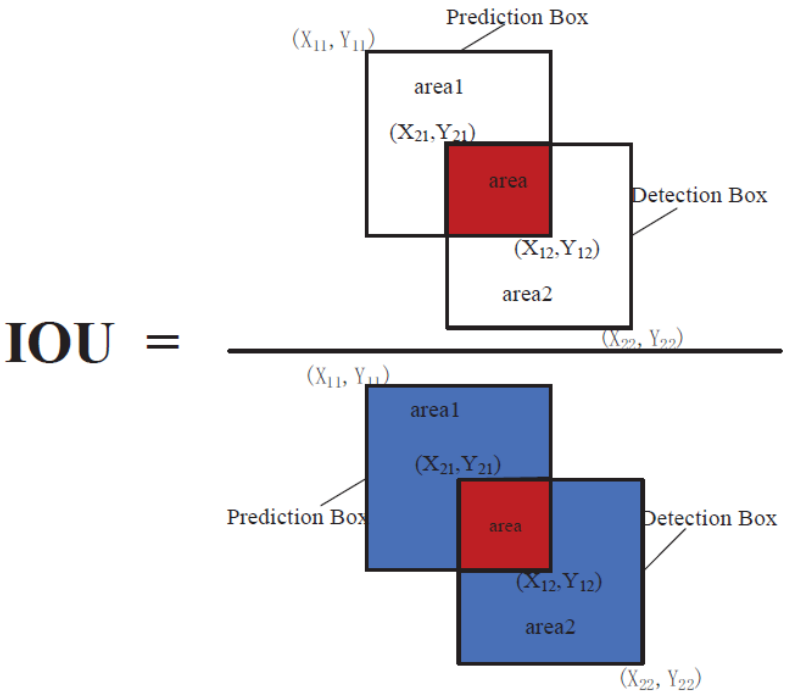

Fig. 5.1 Intersection-Over-Union and Bounding box coordinates [85]

5.3. METHODOLOGY

The first step in this approach was to deploy an object detection algorithm to provide us with the loosely detected bounding boxes for the input image given. For the object detection work, this research work uses Faster R-CNN [65] as its algorithm. There is no specific reason to choose this specific algorithm but the simplicity with which the

output of the bounding box is generated. Since the PASCAL VOC dataset generally uses 4 coordinates of the bounding box as x_{min}, x_{max}, $y_{min,}$ and y_{max} it was comparatively easier to implement.

All the hyperparameters mentioned in the original implementation were kept the same except for the following. When being trained on the original dimensions of the fully connected layer and VGG, since the output feature map size of VGG is 7 × 7 × 512, it comes out at 25088. Along with that, the number of fully connected layers that are connected at the end was dimension 4098. This implementation actually caused an out-of-memory error for the machine on which it was trained which in this case is a laptop with GTX1650 and 4GB of graphics memory. So to circumvent this, a workaround in the form of reducing the number of units in the fully-connected layers was done. Multiple approaches and different architecture uses were done to get the best possible and closest mAP value to the original implementation of 69.9.

After the calculation of bounding box coordinates, the coordinates were then converted into PASCAL VOC-like annotation and which will be used for both training as well testing. A total of 1000 images containing a mixed number of images were picked up with the presence of other items in the images too. The agent in the DQN implementation was trained on 200 images and then was tested on 200 images for various epochs but the optimal epoch came around at 80. Here the DQN agent follows an epsilon greedy approach when trying to get the best possible reward. The DQN architecture and the reward function were modified from the previous implementation because of the reason that when the original implementation was used it used just one non-overlapping instance of the objects. So in this the DQN used had 4 fully connected layers, two of the layers contained 2000 neurons and the last two of them contained 500 neurons each with the last later being a linear activated layer with 9 outputs because of the 9 possible actions the agent can take. The modified reward movement is given as:

$$r(a_t) = \begin{cases} 4, & if\ IoU_t > IoU_{t-1} \\ -10, & otherwise \end{cases} \tag{5.1}$$

The modified reward trigger is given by:

$$r(a_t) = \begin{cases} +r_1 + c \times \Gamma^2, & \text{if } IoU_t > \beta \\ -r_2, & otherwise \end{cases} \tag{5.2}$$

The value of $\Gamma = \frac{10-t}{10}$, $r_1 = 5$, $r_2 = 12$, $c = 5$. The value β is the specified threshold for the intersection-over-union of the agent to look for when multiple boxes are taken into

consideration. The a_1 calculates the amount of box's current dimension that is added or removed from its coordinates. There is a chance of over-scaling of the bounding boxes and thus a_2 is set to $\frac{a_1}{2}$. The list of actions that the agent was allowed to take is given in the table below:

Table. 4.1 Different action steps the agent takes

Action Steps	Corresponding Equation	Action Steps	Corresponding Equation
right	$y_{min} + a_1 \times width$ $y_{max} + a_1 \times width$	left	$y_{min} - a_1 \times width$ $y_{max} - a_1 \times width$
up	$x_{min} - a_1 \times height$ $x_{max} - a_1 \times height$	down	$x_{min} + a_1 \times height$ $x_{max} + a_1 \times height$
fat	$x_{min} + a_2 \times height$ $x_{max} - a_2 \times height$	thin	$y_{min} + a_2 \times width$ $y_{max} - a_2 \times width$
wide	$y_{min} - a_2 \times width$ $y_{max} + a_2 \times width$	tall	$x_{min} - a_2 \times height$ $x_{max} + a_2 \times height$

Here the height and width refer to the input image dimensions being fed into the feature extraction architecture of the implementation. Different iterations from VGG and ResNet family were used as the backbone for feature extraction.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1. MODIFIED Faster R-CNN IMPLEMENTATION

The modified approach for Faster R-CNN where done and below are the various approaches to the experiments.

1) Approach 1: 1024 nodes in the 2 FC layers and using pre-trained ImageNet weights and with shuffling and for 8 epochs:

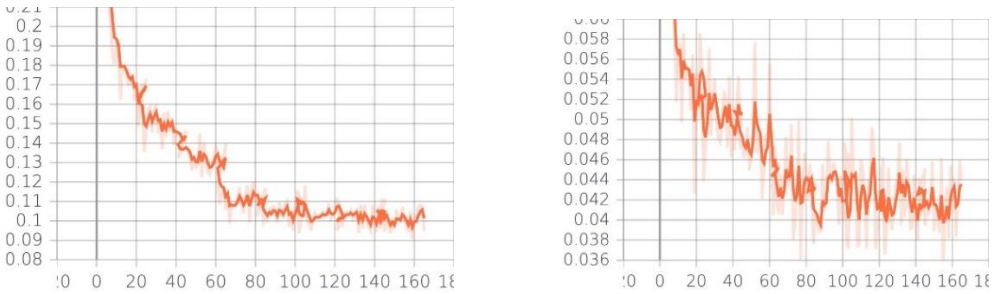

Fig. 6.1.1 RoI and RPN box losses

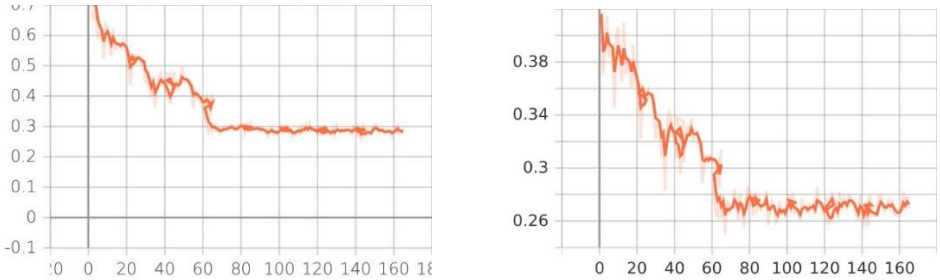

Fig. 6.1.2 RoI and RPN class loss

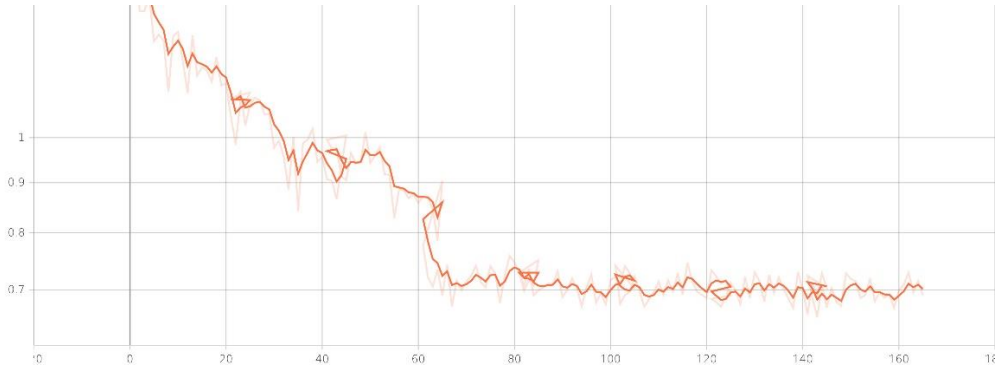

Fig. 6.1.3 Total Loss in the approach

2) Approach 2: 1024 nodes in the 2 FC layers and using pre-trained ImageNet weights, without shuffling, for 8 epochs:

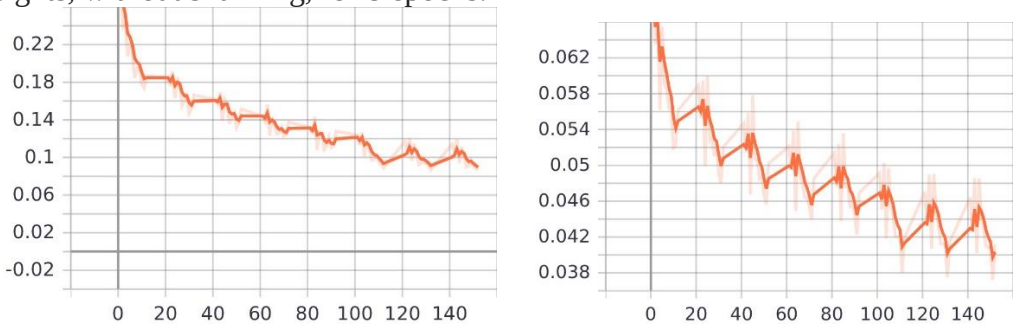

Fig. 6.1.4 RoI and RPN box losses

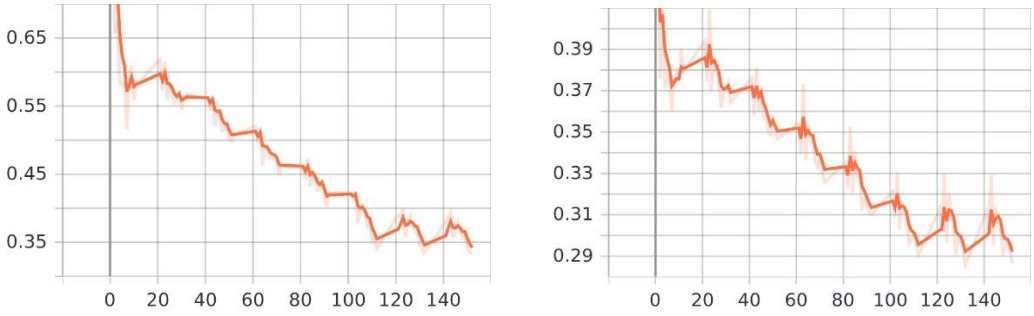

Fig. 6.1.5 RoI and RPN class loss

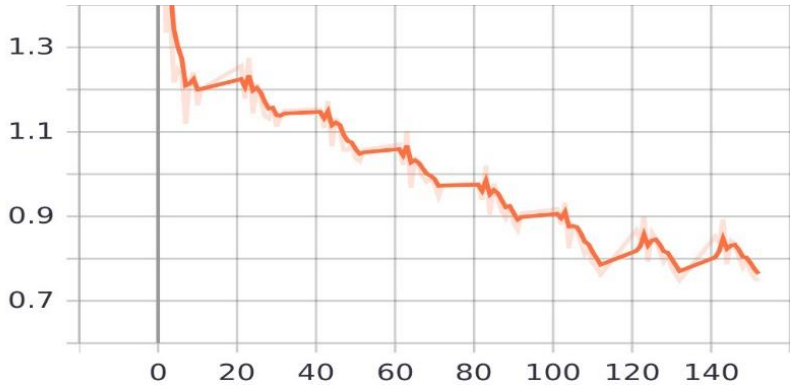

Fig. 6.1.6 Total Loss in the approach

3) Approach 3: 1024 nodes in the 6 FC layers and using pre-trained ImageNet weights, with shuffling, for 8 epochs and dropout of 0.50:

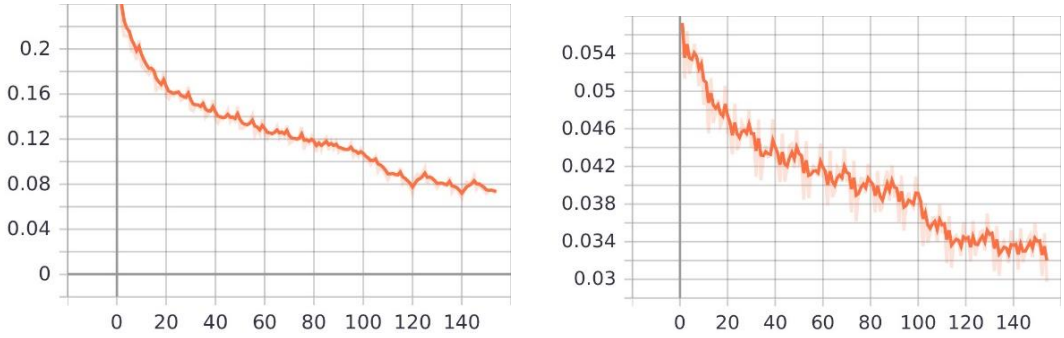

Fig. 6.1.7 RoI and RPN box losses

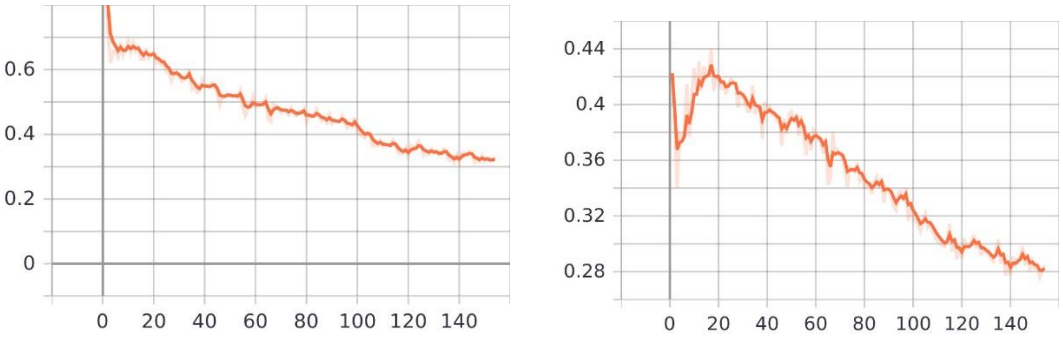

Fig. 6.1.8 RoI and RPN class loss

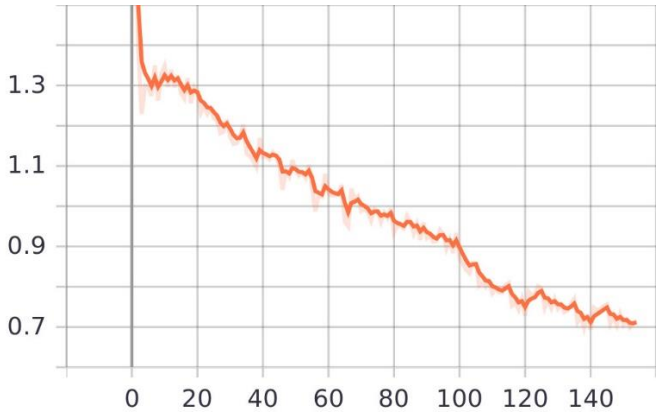

Fig. 6.1.9 Total Loss in the approach

4) Approach 3: 1024 nodes in the 3 FC layers and using pre-trained ImageNet weights, with shuffling, for 12 epochs and dropout of 0.50, unfreezed all layers:

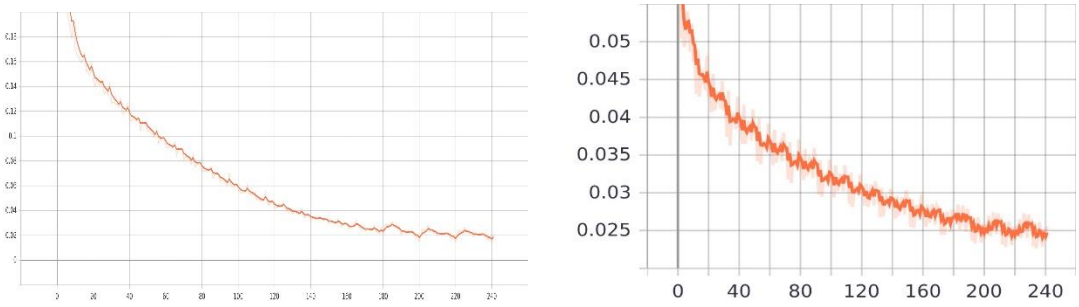

Fig. 6.1.10 RoI and RPN box losses

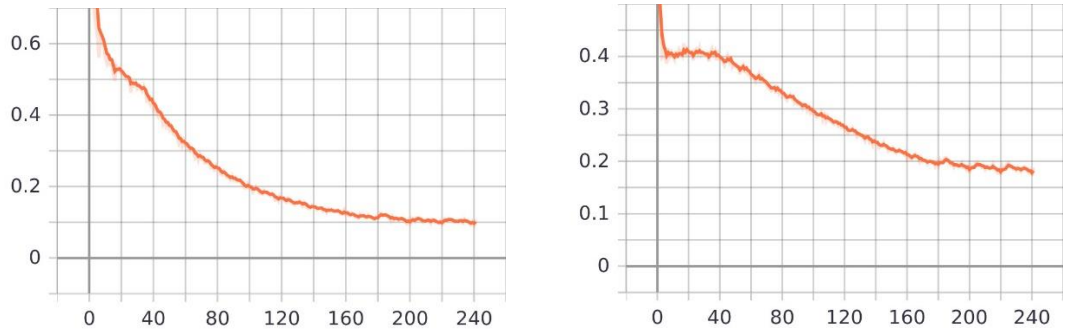

Fig. 6.1.11 RoI and RPN class loss

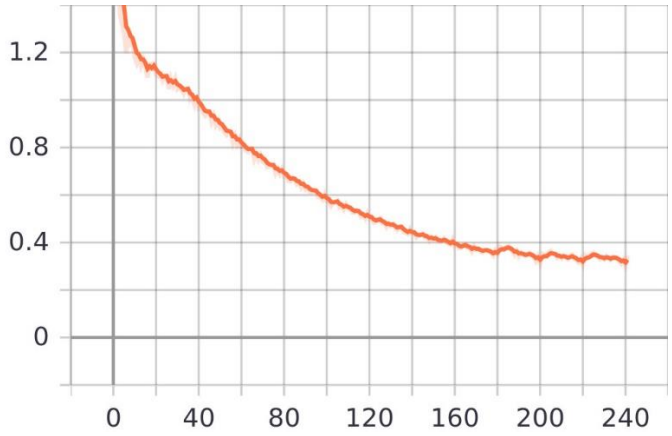

Fig. 6.1.12 Total Loss in the approach

From the above images we can conclude that the best result came out for the last approach when all the layers were un-freezed and 3 dense layers were used. In all the training initial learning rate was kept at 5×10^{-4} with the Adam optimizer and after 8 epochs the learning rate was reduced to 1×10^{-5}.

6.2. DQN IMPLEMENTATION

The IoU threshold for the DQN implementation were set at 0.50 and 0.80 andthe number of epochs were varied in between 50 and 80. The best correction was available on 80 epochs.

1) Approach 1: 50 Epochs, 0.50 IoU Threshold:

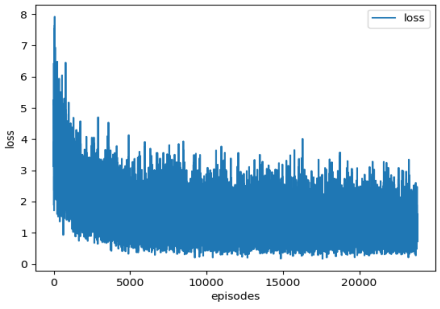

Fig. 6.2.1 IoU Loss Plot

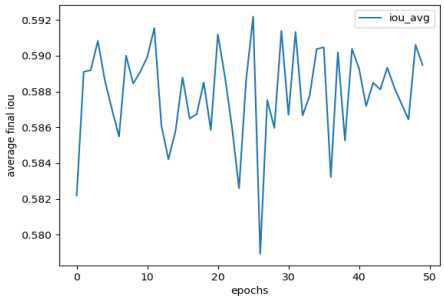

Fig. 6.2.2 Average IoU after correction

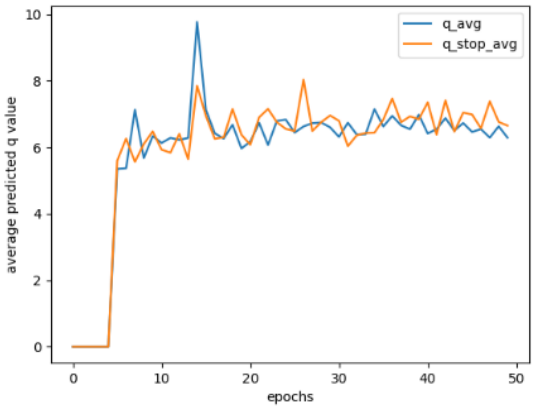

Fig. 6.2.3 Average predicted Q-Value

2) Approach 2: 50 Epochs, 0.80 IoU Threshold:

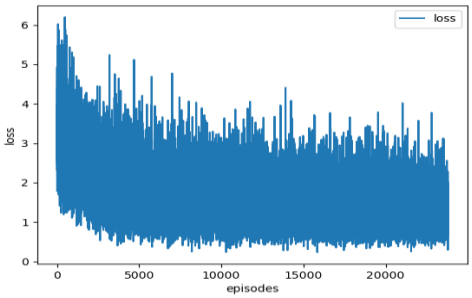

Fig. 6.2.4 IoU Loss Plot

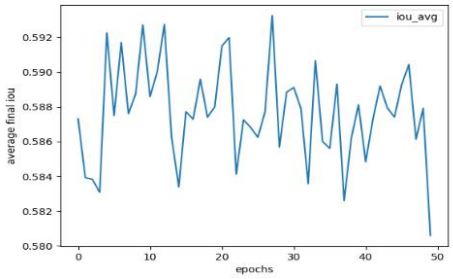

Fig. 6.2.5 Average IoU after correction

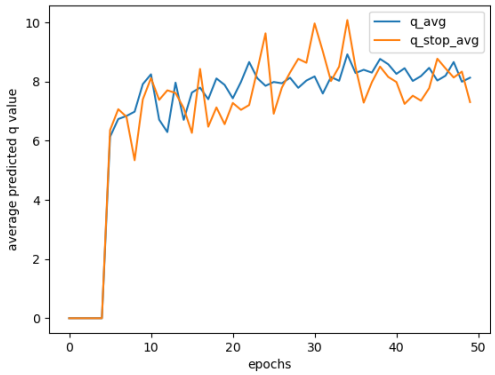

Fig. 6.2.6 Average predicted Q-Value

3) Approach 3: 80 Epochs, 0.50 IoU Threshold:

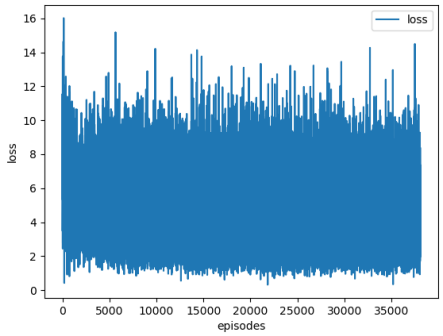

Fig. 6.2.7 IoU Loss Plot

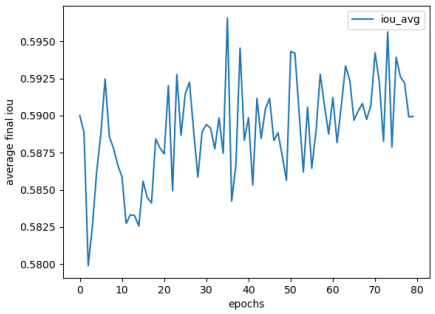

Fig. 6.2.8 Average IoU after correction

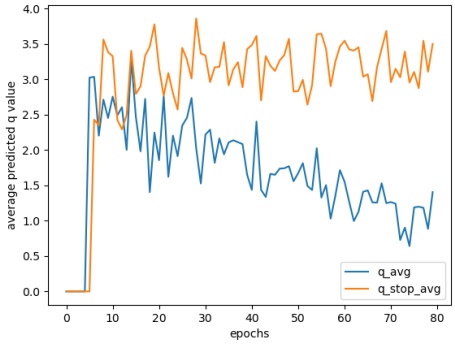

Fig. 6.2.9 Average predicted Q-Value

Table 1. mAP values corresponding to modified implementation Faster R-CNN

Model Architecture	mAP (PASCAL VOC 2007)
2 FC Layers, 1024, 8 Epochs, Shuffling	34.6
2 FC Layers, 1024, 8 Epochs	34.8
6 FC Layers, 1024, 50% Dropout	38.6
3 FC Layers, 1024, 50% Dropout	62.4

Table 2. AP values corresponding to DQN agent refinement

Model Architecture	Initial AP@50	Final AP@50
VGG19, 50 Epoch, IoU@50	69.93	70.12
VGG19, 50 Epoch, IoU@80	69.93	65.23
ResNet50, 80 Epoch, IoU@50	69.93	72.22

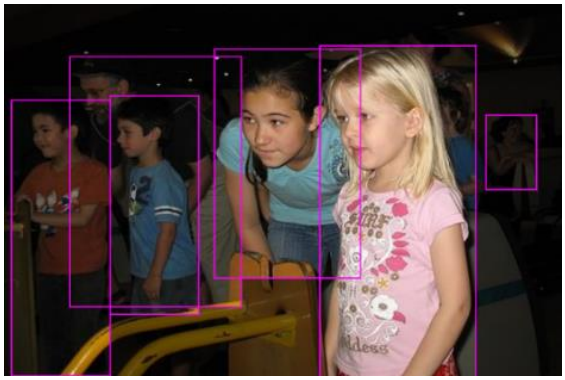

Fig. 6.2.10 Ground Truth Annotations

Fig. 6.2.11 Faster R-CNN Detections

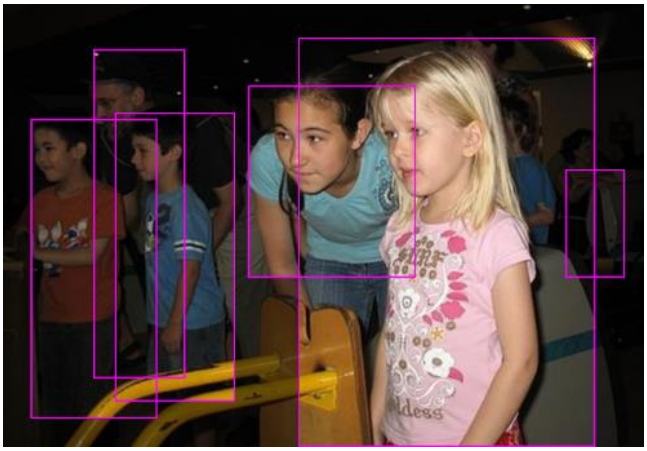

Fig. 6.2.12 Corrected Bounding Box by DQN Agent

From the above results and discussions, one can draw the conclusion that even though the detections might be weaker, the DQN agent because of its high negative reward learns that if it degrades the box to lower IoU it will be penalized. As in the above table, it actually degrades good bounding box detections to a lower AP. The AP calculated is based on a single class chosen from the PASCAL VOC dataset, namely “person” class because of its ability to contain overlapped information and since here the only consideration is a single class, instead of mAP for complete dataset, AP is used and calculated using an approach mentioned in [86]. The implementation of Faster R-CNN is mainly based on [87] with usage from other blogs and repositories like [88] and [89].

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

So from the above results, we can thus conclude that the DQN agent was indeed able to correct the incorrect bounding boxes that were originally detected by the object detection algorithm. The improvement though not much but around 3% because the object detection algorithm was trained good enough to find out the bounding boxes. Since this work is aimed at correcting bounding boxes with the comparatively lesser hassle, this work still can be improved mainly with the use of better state-of-the-art datasets like COCO, Open Image Dataset by Google, and much more. The reward function isn't perfect but was able to perform well for this specific work and dataset. One possible research area can be for approaches like this proposing a general reward function. One of the avenues which inhibited the exploration was a publicly available annotated dataset of any warehouse around the world. This work can find it's was mainly in warehouses around the world for inventory updates, damage updates, and more. Whenever the organization wants to change the target object they can change the DQN on a smaller dataset with the updated object, which might in turn be able to reduce the amount of human labor and currency. Future work can also be done in the area of using a continuous action approach for the agents. Here the policy used by the DQN agent is epsilon greedy, a different policy approach can be designed for the agent to make use of.

REFERENCES

[1] Jacobson, Doug, 2019. *Iowa State University*. [Online] Available at:https://theconversation.com/what-was-the-first-computer-122164#:~:text=The%20first%20mechanical%20computer%2C%20The,computer%20we%20all%20use%20today.&text=The%20ABC%20weighed%20over%20700,had%20small%20capacitors%20on%20it[accessed 21 May 2022].

[2] Das, K. and Behera, R.N., 2017. A survey on machine learning: concept, algorithms and applications. *International Journal of Innovative Research in Computer and Communication Engineering*, vol. *5, no 2*, pp.1301-1309.

[3] Hebb, D.O., 2005. *The organization of behavior: A neuropsychological theory*. Psychology Press.

[4] Turing, Alan M. "Computing machinery and intelligence." In *Parsing the turing test*, pp. 23-65. Springer, Dordrecht, 2009.

[5] Thota, Subash, 2017. *Synetics.* [Online] Available at:https://www.smdi.com/the-evolution-of-machine-learning/ [accessed 21 May 2022].

[6] McCarthy, J., Minsky, M.L., Rochester, N. and Shannon, C.E., 2006. A proposal for the Dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, vol. *27, no. 4*, pp.12-12.

[7] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, vol. 65, no.* 6, pp. 386–408.

[8] T. Cover and P. Hart, "Nearest neighbor pattern classification," in *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21-27, January 1967.

[9] Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. *Nature* **323, pp.** 533–536 , 1986.

[10] Y. LeCun *et al*., "Backpropagation Applied to Handwritten Zip Code Recognition," in *Neural Computation*, vol. 1, no. 4, pp. 541-551, Dec. 1989.

[11] Sah, S., 2020. Machine learning: a review of learning types.

[12] Sarker, I.H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN COMPUT. SCI.* **2,** 160 , 2021.

[13] Nasteski, V., 2017. An overview of the supervised machine learning methods. *Horizons. b*, *4*, pp.51-62.

[14] Sandhya N. dhage, Charanjeet Kaur Raina. (2016) *A review on Machine Learning Techniques*. In International Journal on Recent and Innovation Trends in Computing and Communication, Volume 4 Issue 3.

[15] Osisanwo, F.Y., Akinsola, J.E.T., Awodele, O., Hinmikaiye, J.O., Olakanmi, O. and Akinjobi, J., 2017. Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, vol. *48*, *no.* 3, pp.128-138.

[16] Mladenić, Dunja, Janez Brank, Marko Grobelnik, and Natasa Milic-Frayling. "Feature selection using linear classifier weights: interaction with classification models." In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 234-241. 2004.

[17] Rish, Irina. "An empirical study of the naive Bayes classifier." In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, pp. 41-46. 2001.

[18] Kleinbaum, D.G., Dietz, K., Gail, M., Klein, M. and Klein, M., 2002. *Logistic regression* (p. 536). New York: Springer-Verlag.

[19] Rosenblatt, Frank. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[20] Baum, E.B., 1988. On the capabilities of multilayer perceptrons. *Journal of complexity*, vol. 4, no. 3, pp.193-215.

[21] Pisner, Derek A., and David M. Schnyer. "Support vector machine." In *Machine learning*, pp. 101-121. Academic Press, 2020.

[22] Kotsiantis, S.B., 2013. Decision trees: a recent overview. *Artificial Intelligence Review*, vol. 39, no. 4, pp.261-283.

[23] Biau, G. and Scornet, E., 2016. A random forest guided tour. *Test*, vol. 25, no. 2, pp.197-227.

[24] Jain, A.K., Mao, J. and Mohiuddin, K.M., 1996. Artificial neural networks: A tutorial. *Computer*, vol. 29, no.3, pp.31-44.

[25] Friedman, N., Geiger, D. and Goldszmidt, M., 1997. Bayesian network classifiers. *Machine learning*, vol. 29, no. 2, pp.131-163.

[26] Osisanwo, F.Y., Akinsola, J.E.T., Awodele, O., Hinmikaiye, J.O., Olakanmi, O. and Akinjobi, J., 2017. Supervised machine learning algorithms: classification and

comparison. *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp.128-138.

[27] Celebi, M. Emre, and Kemal Aydin, eds. *Unsupervised learning algorithms*. Berlin: Springer International Publishing, 2016.

[28] Zhang, J., Chen, W., Gao, M. and Shen, G., 2017. K-means-clustering-based fiber nonlinearity equalization techniques for 64-QAM coherent optical communication system. *Optics express*, vol. 25, no. 22, pp.27570-27580.

[29] Guo, Gongde, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. "KNN model-based approach in classification." In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 986-996. Springer, Berlin, Heidelberg, 2003.

[30] Murtagh, F. and Contreras, P., 2012. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp.86-97.

[31] Abdi, H. and Williams, L.J., 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp.433-459.

[32] Baker, K., 2005. Singular value decomposition tutorial. *The Ohio State University*, *24*.

[33] Stone, James V. "Independent component analysis: an introduction." *Trends in cognitive sciences* 6, no. 2, pp. 59-64, 2002.

[34] Atallah, D.M., Badawy, M., El-Sayed, A. and Ghoneim, M.A., 2019. Predicting kidney transplantation outcome based on hybrid feature selection and KNN classifier. *Multimedia Tools and Applications*, vol. 78, no. 14, pp.20383-20407.

[35] Arulkumaran, K., Deisenroth, M.P., Brundage, M. and Bharath, A.A., 2017. A brief survey of deep reinforcement learning. *arXiv*.[Online]. https://arxiv.org/abs/1708.05866

[36] Müller, Berndt, Joachim Reinhardt, and Michael T. Strickland. *Neural networks: an introduction*. Springer Science & Business Media, 1995.

[37] McCulloch, W.S. and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), pp.115-133.

[38] Jean-Christophe B. Loiseau, 2019. Towards Data Science. [Online]. https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a. [accessed 21 May 2022].

[39] Widrow, Bernard, and Marcian E. Hoff. *Adaptive switching circuits*. Stanford Univ Ca Stanford Electronics Labs, 1960.

[40] Minsky, M.L. and Papert, S.A., 1988. Perceptrons: expanded edition, ACM.

[41] Werbos, Paul. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science. Thesis (Ph. D.). Appl. Math. Harvard University.

[42] Linnainmaa, S., 1976. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, vol. 16, no. 2, pp.146-160.

[43] Wythoff, B.J., 1993. Backpropagation neural networks: a tutorial. *Chemometrics and Intelligent Laboratory Systems*, vol. 18, no. 2, pp.115-155.

[44] Kunihiko Fukushima (1980). *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.* , vol. 36, no. 4,pp. 193–202.

[45] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, vol. 1, no. 4, pp.541-551.

[46] LeCun, Y., 1989. Generalization and network design strategies. *Connectionism in perspective*, *19*(143-155), p.18.

[47] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. and Berg, A.C., 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, vol. 115, no. 3, pp.211-252.

[48] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

[49] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." In *Icml*. 2010.

[50] Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." In *European conference on computer vision*, pp. 818-833. Springer, Cham, 2014.

[51] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. 2015.

[52] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv. preprint arXiv:1409.1556* (2014). [Online]. https://arxiv.org/abs/1409.1556

[53] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

[54] Srivastava, Rupesh Kumar, Klaus Greff, and Jürgen Schmidhuber. "Highway networks." *arXiv preprint arXiv:1505.00387* (2015). [Online]. https://arxiv.org/abs/1505.00387

[55] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity mappings in deep residual networks." In *European conference on computer vision*, pp. 630-645. Springer, Cham, 2016.

[56] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1. IEEE, 2001, pp. 1–1

[57] Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object detection in 20 years: A survey." *arXiv preprint arXiv:1905.05055* (2019).[Online]. https://arxiv.org/abs/1905.05055

[58] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1. IEEE, 2005, pp. 886–893

[59] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008, pp. 1–8

[60] Sermanet, Pierre, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. "Overfeat: Integrated recognition, localization and detection using convolutional networks." *arXiv preprint arXiv:1312.6229* (2013).[Online]. https://arxiv.org/abs/1312.6229

[61] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580– 587.

[62] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders, "Segmentation as selective search for object recognition," in Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011, pp. 1879–1886.

[63] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in European conference on computer vision. Springer, 2014, pp. 346–361.

[64] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[65] 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).

[66] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[67] Everingham, M., Van Gool, L., Williams, C.K.I. *et al.* The PASCAL Visual Object Classes (VOC) Challenge. *Int J Comput Vis* **88,** 303–338 (2010).

[68] Thorndike, E. L. (1898). Animal intelligence: An experimental study of the associative processes in animals. *The Psychological Review: Monograph Supplements*, vol. 2, no. 4, i–109.

[69] Thorndike, Edward L. "The Law of Effect." *The American Journal of Psychology*, vol. 39, no. 1/4, 1927, pp. 212–22. *JSTOR*.

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

[71] Bellman, R., 1966. Dynamic programming. *Science*, *153*(3731), pp.34-37.

[72] Bellman, R., 1957. A Markovian decision process. *Journal of mathematics and mechanics*, pp.679-684.

[73] Howard, R. A. (1960). *Dynamic programming and Markov processes.* John Wiley.

[74] Arthur, S., 1959. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, vol. 3, no. 3, pp.210-229.

[75] Minsky, M., 1961. Steps toward artificial intelligence. *Proceedings of the IRE*, vol. 49, no. 1, pp.8-30.

[76] Minsky, Marvin Lee. *Theory of neural-analog reinforcement systems and its application to the brain-model problem*. Princeton University, 1954.

[77] Watkins, Christopher John Cornish Hellaby. "Learning from delayed rewards." (1989). PhD Thesis.

[78] Watkins, C.J. and Dayan, P., 1992. Q-learning. *Machine learning*, *8*(3), pp.279-292.

[79] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M.G. and Pineau, J., 2018. An Introduction to Deep Reinforcement Learning. *Foundations and Trends in Machine Learning*, vol. 11, no. 3-4, pp.219-354.

[80] Bellman, R. E. and S. E. Dreyfus. 1962. "Applied dynamic programming". Princeton University Press.

[81] Li, Y., 2017. Deep reinforcement learning: An overview. *arXiv preprint.* [Online]. https://arxiv.org/abs/1701.07274

[82] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*. [Online]. https://arxiv.org/abs/1312.5602

[83] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *Nature*, Vol. 518, no. 7540, pp.529-533.

[84] Ayle, Morgane, Jimmy Tekli, Julia El-Zini, Boulos El-Asmar, and Mariette Awad. "BAR—A Reinforcement Learning Agent for Bounding-Box Automated Refinement." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 2561-2568. 2020.

[85] Cheng, S., Zhao, K. and Zhang, D., 2019. Abnormal Water Quality Monitoring Based on Visual Sensing of Three-Dimensional Motion Behavior of Fish. *Symmetry*, vol. 11, no. 9, p.1179.

[86] ZFTurbo- https://github.com/ZFTurbo/Mean-Average-Precision-for-Boxes.

[87] Original Python Implementation of Faster R-CNN: https://github.com/rbgirshick/py-faster-rcnn

[88] Faster R-CNN on Open Image Dataset by Google: https://github.com/RockyXu66/Faster_RCNN_for_Open_Images_Dataset_Keras

[89] Another experimental implementation of Faster R-CNN: https://github.com/smallcorgi/Faster-RCNN_TF

PUBLICATIONS

1. **IEEE International Conference on "Machine Learning, Big Data, Cloud and Parallel Computing: Trends, Perspectives and Prospects" (Com-IT-Con-2022), 26th-27th May 2022, Manav Rachna International Institute of Research & Studies, Faridabad, India**

   Paper ID: 237
   Paper Title: PREDICTING GROWTH IN TOURISM INDUSTRY USING MACHINE LEARNING METHODS
   Authors: Anindya Ghosal, Jyotsna Singh, Ashutosh Singh
   Status: Accepted and Presented

2. **2nd International Conference on Signals, Machines, and Automation *(SIGMA)*-2022, 05th – 06th August 2022, Netaji Subhas University of Technology, New Delhi, India**

   Paper ID: 63
   Paper Title: Bounding Box Refinement Agent for Overlapping Objects
   Authors: Anindya Ghosal, Jyotsna Singh, Ashutosh Singh
   Status: Under Review