Application of Machine Learning in Communication Networks

M.Tech. Thesis

By

SHUBHAM GUPTA

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JUNE 2022

Application of Machine Learning in Communication Networks

A THESIS

Submitted in partial fulfillment of the requirements for the award of the degree

of

Master of Technology

by

SHUBHAM GUPTA

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JUNE 2022

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Application of Machine Learning in communication Networks** in the partial fulfillment of the requirements for the award of the degree of MASTER OF TECHNOLOGY and submitted in the **Department of Electrical Engineering**, Indian Institute of Technology Indore, is an authentic record of my own work carried out during the time period from AUGUST 2020 to JUNE 2022 under the supervision of Prof. Vimal Bhatia, Professor, Department of Electrical Engineering, IIT Indore.
The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Shubham Gupta
31/05/2022

Signature of the student with date
Shubham Gupta

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

[illegible]
31.05.2022
Signature of the Supervisor of
M.Tech. thesis (with date)
Prof. Vimal Bhatia

Shubham Gupta has successfully given his/her M.Tech. Oral Examination held on **06/06/2022**.

[illegible]
06.06.2022
Signature(s) of the Supervisor(s) of M.Tech. thesis
Date:...../06/2022

[illegible]
Convener, DPGC
Date:06/06/2022

[illegible]
06.06.2022
Signature of PSPC (**Dr. V. Kanhangad**)
Date:...../06/2022

Signature of PSPC (**Prof. D. K. Rai**)
Date:...../06/2022

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my thesis supervisors Prof. Vimal Bhatia for their constant support, patience, encouragement, and immense knowledge. Their guidance helped me in conducting the research and writing of this M.Tech thesis. I could not have imagined having a better advisor and mentor for my masters study.

Besides my advisor, I would like to thank my PSPC members: Prof. Dhirendra Kumar Rai and Dr. Vivek Kanhangad for their encouragement and insightful comments. Their questions and suggestions helped me to widen my knowledge from various perspectives. I also would link to thank IIT Indore for all the facilities and Ministry of Education for financial support.

Many thanks to Ms.Purva Sharma for her coopertion in the project and helping in writting this thesis.

I thank my fellow labmates and cohort members in Signals and Software Group: Anupma Sharma, Krishnendu S, Vaishali Sharma, Shubham Bisen, Abhinav Singh Parihar, Deepak Kumar, Justin Jose, Vidhya Bhaskar Shukla, Parvez Sheikh, Nikhil Kaler, Bhagyashree Gour, Siddhesh Gawali for the stimulating discussions, their support and all the fun we have had in the last year.

Lastly, I also thank my family and friends for their unceasing encouragement and moral support throughout this journey.

Shubham Gupta
M.Tech.
(Communication and Signal Processing)
2002102012
Department of Electrical Engineering,
IIT Indore

ABSTRACT

Human curiosity has always led to the development of technologies. To mimic the human brain in machines, led to the development of Artificial Intelligence (AI). The aim is to develop a general AI which can learn multiple complex tasks and make decisions. Major work that has been done till now falls in the category of artificial narrow intelligence (ANI), where multiple single tasks have been mastered like object detection, speech to text, predictions, etc. All this is made possible by machine learning (ML) and deep learning (DL) techniques that learn from huge labeled data source which is abundantly available in this era.

Reinforcement learning (RL) is a different learning technique than the previous ML methods. It is an experience based learning which uses rewards and punishment to build an optimal policy, just like a human baby learns. Combining DL and RL techniques together form a very powerful method called deep reinforcement learning (DRL). This is the initial step in the artificial general intelligence (AGI) category.

In this thesis we have explored the DRL technique for optimization of routing and resource allocation in quantum key distribution (QKD) networks in optical networks. DRL has been successful in serving the purpose in simulated environments, thus improve the resource utilization and reduce the blocking of requests in the QKD network as compared to the baseline algorithms.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
AGI	Artificial General Intelligence
ANI	Artificial Narrow Intelligence
ASI	Artificial Superficial Intelligence
CCh	Classical Channel
CV	Computer Vision
DCh	Data Channel
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
EDA	Exploratory Data Analysis
GAN	Generative Adversarial Networks
GPU	Graphical Processing Unit
ML	Machine Learning
MLP	Multi Layer Perceptron
NN	Neural Network
NLP	Natural Language Processing
PF	Polarization Filter
QCh	Quantum Channel
QKD	Quantum Key Distribution
QSS	Quantum Signal Source
RL	Reinforcement Learning
RNG	Random Number Generator
SDN	Sotware Defined Network
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TPU	Tensor Processing Unit

Chapter 1

Introduction

1.1 Introduction to Artificial Intelligence

God's creation has always intrigued human curiosity which led us to discover the governing laws and design of how this universe works. In the same line, we keep on trying to understand the mechanism of the most sophisticated chemical factory "The human body" about how our body communicates inside, takes input through our sensory organ and ultimately makes decisions and learns new things.

"Brain" is a system which is responsible for our ability of reasoning, use of complex language, to solve difficult problems, and do introspection. Thus, brain is one of the most interesting subjects to understand and simulate its functionality. The aim is to build a similar system which can be used along with different machinery and robots so that they can also perform similar task as brain does. That's what we term as Artificial Intelligence (AI) [1].

Different streams of science are working to mimic human intelligence through several means. In electrical engineering and computer science, we are leveraging the power of heavy computational hardware resources like GPUs and TPUs to accomplish this task by processing huge data sources and running complex algorithms (which may have taken months and years in the previous decade) to fulfill this task. Fortunately, we have come a long way towards AI and are able to perform many tasks successfully as humans do and in some cases better than humans [1].

Evolution of AI: Over time AI has gained extensive attention in the field of various real-time applications. Hence, development in the field of AI has been very rapid in last decade. Various organizations such as Google, Facebook, Microsoft, and Tesla has used AI for developing many products which help millions of people over the world.

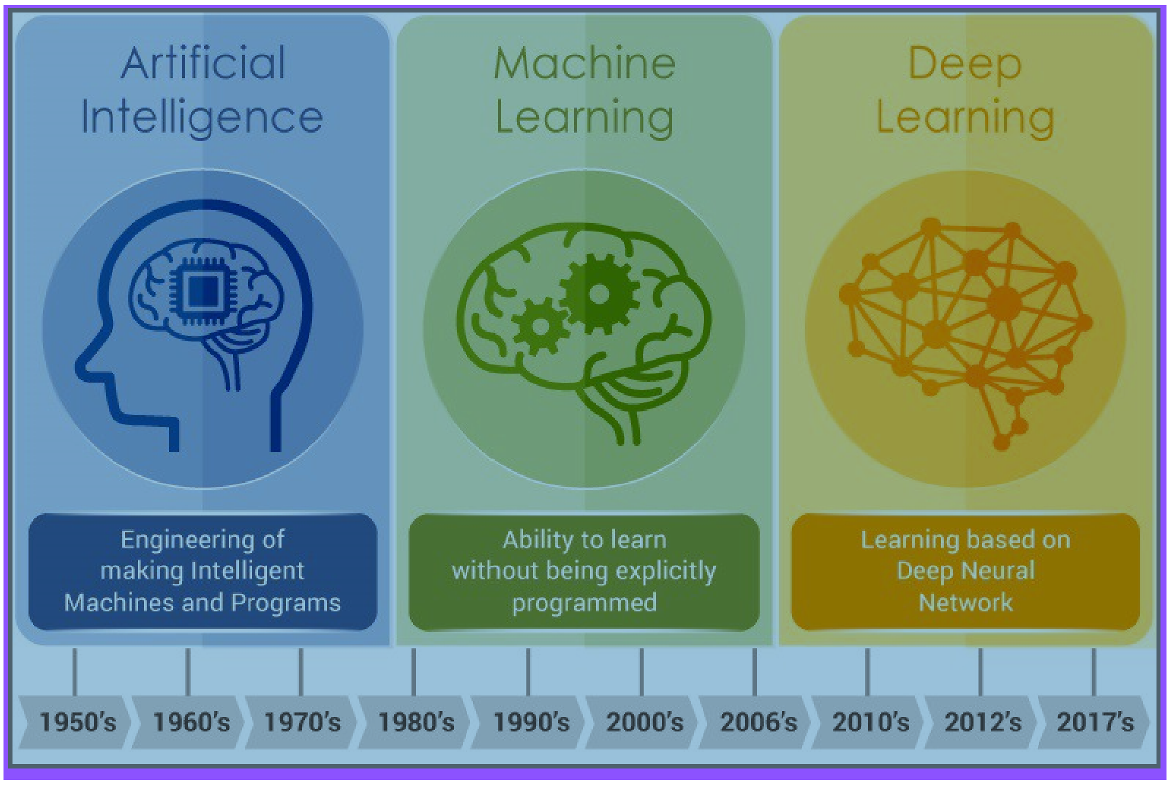

Artificial Intelligence
Machine Learning
Deep Learning
Engineering of making Intelligent Machines and Programs
Ability to learn without being explicitly programmed
Learning based on Deep Neural Network
1950's
1960's
1970's
1980's
1990's
2000's
2006's
2010's
2012's
2017's

Figure 1.1: Evolution of AI

1.1.1 Levels of AI

We can divide level of AI [2] into three phases.

(a) Artificial Narrow Intelligence

In artificial narrow intelligence (ANI) [2], specific tasks are performed autonomously corresponding to human-like capabilities. ANI includes all the existing intelligent machines till date as they serve only specific tasks like pattern recognition, text to speech, self-driving cars, automated voice generation or classification and clustering problems, etc.

(b) Artificial general intelligence

Artificial General Intelligence (AGI) [2] refers to a machine which can resemble a complete human-like intelligence. Systems which can be multi-functional, able to understand, form connections through different domains, argue, reason, memorize as much as a human does. We have not reached this level of intelligence yet, still Reinforcement learning is a starting step towards AGI where machines are able to learn through experiences. Major breakthroughs we got are in games and robot navigation.

(c) Artificial Superficial Intelligence

Artificial superficial intelligence (ASI) [2] will mark a pinnacle in the AI domain. Basically, this system has the prowess of an organization, or say the whole of humanity, which would perform a variety of tasks with higher accuracy. This capability in a machine is usually seen in fiction movies and books. Many people have fear of AI achieving this kind of superficial capability which may harm humankind in the name of establishing peace and order. Though it is too far to build this kind of machine as it will require huge memory and hardware resources.

1.1.2 Machine Learning

Machine learning (ML) is a field of computer science which facilitates AI capabilities. Thus, ML is a subset of AI [3]. Typically statisticians use the algorithm or mathematical modeling to predict the future value of some entity for example gross domestic product (GDP) growth rate by considering the trends of previous data. Now computer science uses similar statistical learning algorithms to build systems which have the capability of learning and improving from experiences without being explicitly programmed. As opposed to traditional programming where a programmer is required to indicate the rules that must be followed to achieve some task but in ML the computers do them automatically.

Tom Mitchell (1998) defined a well-posed Learning Problem as follows:
"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E" [4]

ML algorithms use data as a learning resource to build a rule so that it can produce output to future inputs. Building a rule-based sequential algorithm is not possible for very complex problems or reaching a human-level decision making. To make the machine learn itself is the possible solution for such problems.

'Data is termed as new oil' as it served as the most important factor in any decision making in this era. ML uses these data to build the relationship between different features and output hence the process is known as training. Algorithms are usually referred to a model, when get trained over sufficient iterations we use that trained model to get output for new input data.

It is said that ML can be applied to any field where data can be crunched into a computer in the form of numbers. Understanding of data and domain of field add a great help in applying these algorithms and getting desired results.

Data we use are mainly in worksheets, images, audio, and signals which is first preprocessed to make it suitable to be fed into ML algorithm. This part is known as exploratory data analysis (EDA) [5].

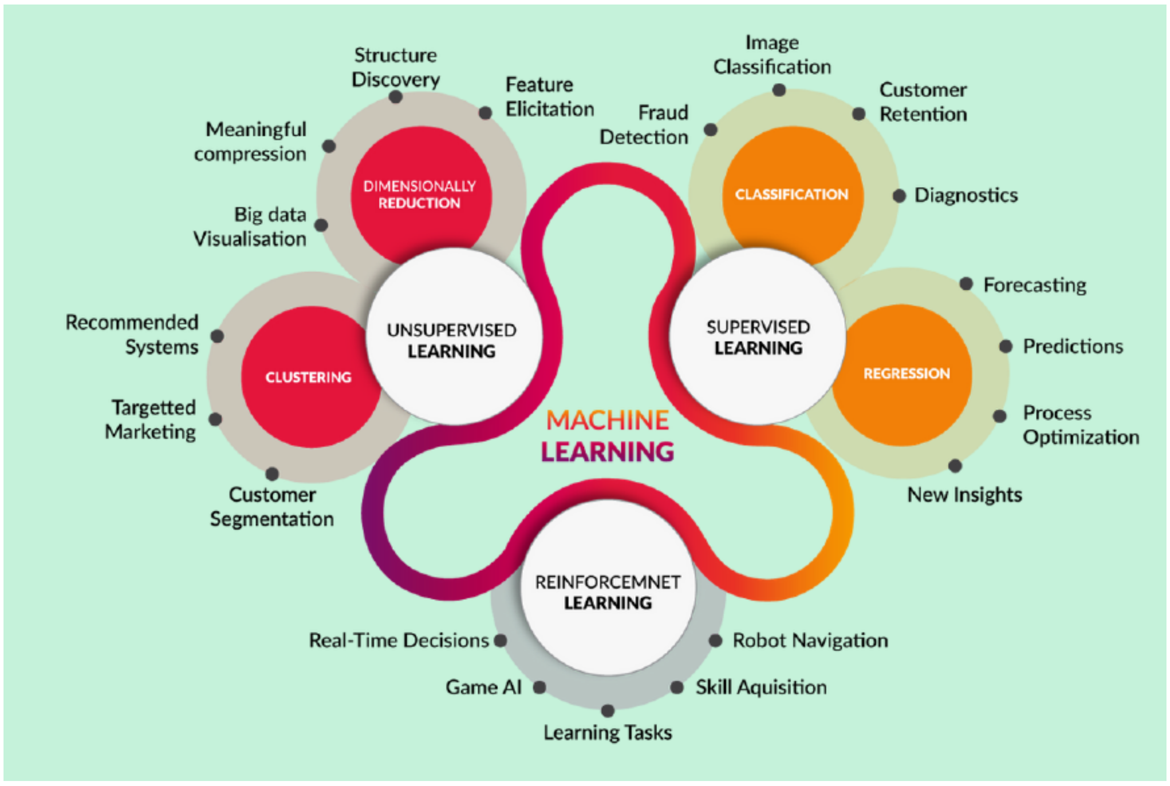


Figure 1.2: Types of Machine Learning

Machine learning is categorized into three types:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

(a) Supervised learning

In supervised learning of learning, the data we use to train our model has labels present in it. Labels represent the output to those inputs, which will be used to build the relationship between input features and output. In this we require human intervention to label the data manually. While training, model fits through the data and adjust its prediction.

Majorly, two types of problems are solved in this learning:

Regression:

- These algorithms are mainly used to predict the output. Like predicting a price of a house considering several features like number of bedrooms, floor area, number of bathrooms, locality etc. Or predicting, how much time will take to commute a certain distance given the model is trained on several affecting factor like traffic, road conditions, weather and so on.
- Forecasting, prediction, getting new insight and process optimization are some of the applications where regression models work.

Classification:

- Another problem that supervised learning helps in is to perform the classification tasks like fraud detection, image classification, and customer retention. Classification algorithm learns the patterns on the basis of data we feed into it.
- In some applications, the accuracy of these algorithms is much better than professional humans in that domain like in the medical field identifying some disease through MRI or X-rays.

(b) Unsupervised Learning

In unsupervised learning, data used for training do not have labels in them. Means we have only input variables present. So our algorithm trains on these data and try to find clusters, similarity inside the data. Major algorithms in unsupervised learning are covered under two categories:

Clustering : Applications covered through this type of algorithm are targeted marketing, customer segmentation, recommended system, collaborative filtering etc. ecommerce website uses these algorithms for recommendation of various products to the customer.

Dimensionality reduction : Applications of dimensionality reduction are image compression, feature elicitation, big data visualization and many more.

(c) Reinforcement learning

Reinforcement learning is an advanced field of ML where the learning process is more similar to what a human baby learns. It involves experience based learning, where an agent continuously interacts with the environment and feedback is generated to guide the agent in the desired direction. Those tasks which are highly dynamic and very complex to create a data based learning like maneuvering a helicopter, playing Go game, or playing atari games are tackled. More details of Reinforcement learning is explained in section 1.1.4. Fig.6 shows the steps of machine learning.

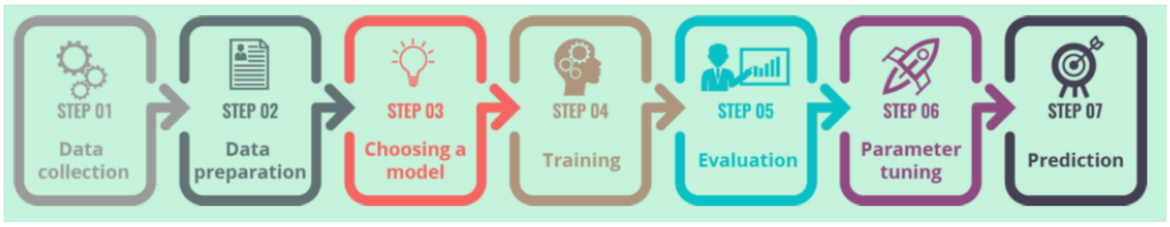


Figure 1.3: Steps of machine learning

Following are the steps of machine learning

- Step1. Data collection: Collection of relevant data is the most important part of any machine learning task. Data should be from authentic source, labeling of data should be done very carefully. Data should be stored securely.

- Step2. Data preparation: After collection of data preparation is done. Preparation involves:

  - Data cleaning: Here missing values are treated, removal of outliers etc.
  - Data transformation: Here different coding methods are used to convert the object variable into numbers. Normalisation of variables is done so that all the values can be reduced to a certain range, generally [0-1].
  - Feature reduction: Features of data are reduced or combined to get better representation of data. Various methods are used to perform this task. Data is divided into test, train, and validation portions. Train and validation data are used while training to learn and check the progress while test data is used to generate metrics to evaluate the performance on unknown data.

- Step3: Model selection: In this step model is selected. Model could be a desired machine learning algorithm or deep learning model. This selection is done with keeping application in mind. Several thumb rules are there to select a model. Model selection depends on data size availability, type of data and so on.

- Step 4: Training: In this we start the training process. Various hyperparameters of models are set, training epochs are decided and many auxiliary functions that support the algorithm are brought to work. Here hardware plays a significant role, sufficient resources must be available to complete the training. Training is done on training set of data and a validation is done on completion of each epoch to track the progress.

- Step 5: Evaluation: Evaluation is done at the end of training to check the result. For evaluation, confusion metrics are generally used to check the accuracy and precision of our model over the data.
- Step6: Parameter tuning: After evaluation we can change the parameters that we had set in training phase to get more accurate results or compare with different settings. Learning rate, number of hidden layers, activation function are examples of such hyperparameters that we can tune to get slightly modified results. On our satisfaction to results we can conclude the experiment and save the model.
- Step 7: Prediction: Saved model is used to get the prediction on incoming new data. Machine learning operations (MLops) methodology is used for continuous training and updation of weights and biases. Website and application are built to fetch the required data and data pipeline methods are used for preprocessing of new data before applying to the trained model.

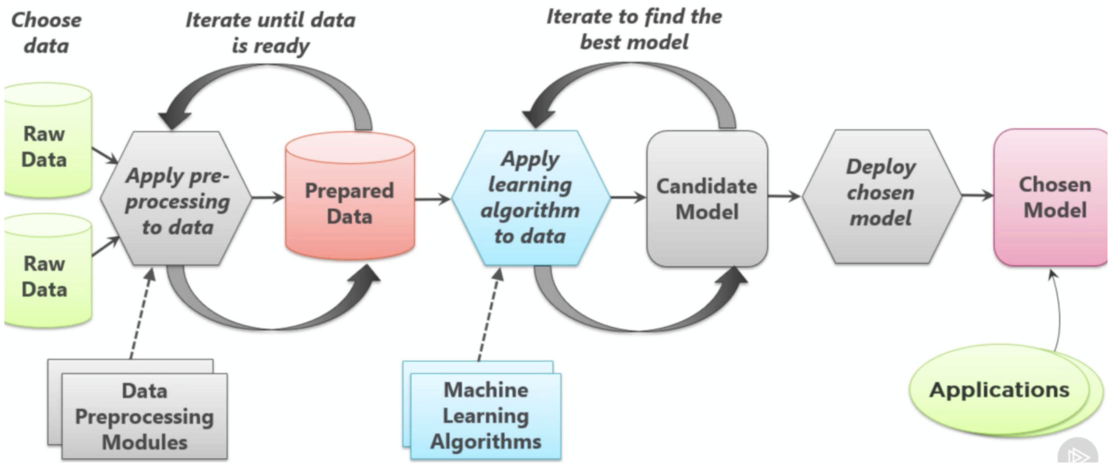

Choose data
Iterate until data is ready
Iterate to find the best model
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications

Figure 1.4: Process of Machine Learning

ML Algorithms:
Various ML algorithms and their categories and use cases are shown in fig.

- Linear Regression
- Decision tree
- Random forest
- Neural network
- Gradient boosting tree
- Kernel Support Vector Machine (SVM)

- Logistic regression
- Naive Bayes
- Xd Boost
- Ensembling

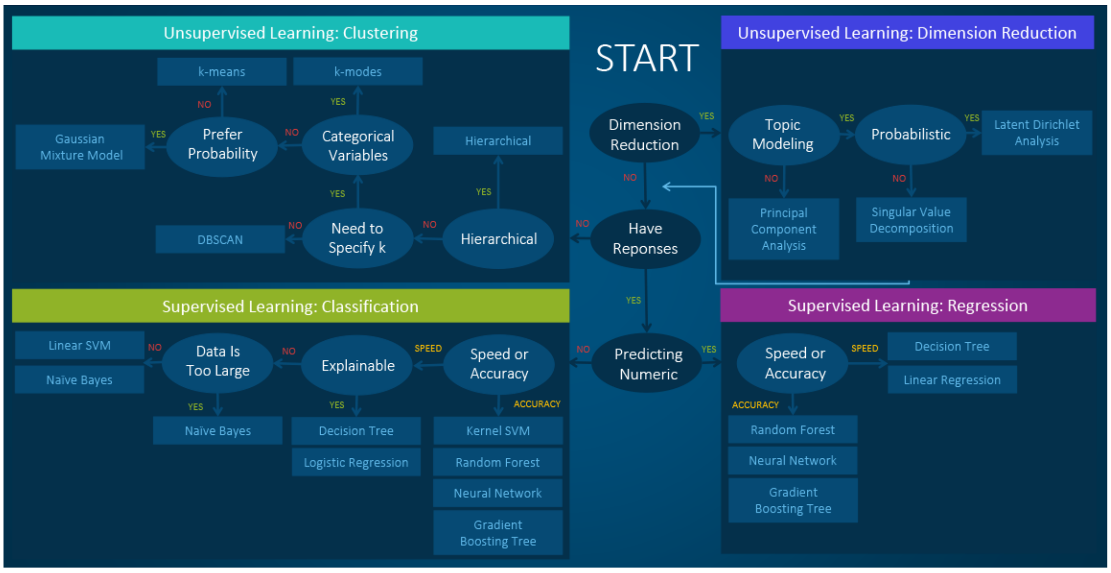


Figure 1.5: Various Machine Learning Algorithms

1.1.3 Deep Learning and Neural Network

As mentioned in the section introduction of AI, the main aim of AI is to mimic the human capabilities of learning and decision making; advancement in strategies has been developed to reach that goal. Though ML algorithms are capable of taking decisions like predicting values and classification tasks, a major issue is, a domain specific person is required to do the feature engineering for satisfactory results. These algorithms are not very useful in computer vision applications involving data in the form of images.

To address this limitation researchers focused on a more general framework which can just take input and perform feature extraction by itself. Neural network made it possible to solve this problem. When a structured layer of neural network is used as an algorithm to learn and take decision process, it is known as Deep learning.

Deep learning is a subset of machine learning, and require large dataset as compared to ML for learning.

Structured layer is termed multi layer perceptron (Mlp) or 'artificial neural network' which can learn and make intelligent decisions. When the number of layers is more than

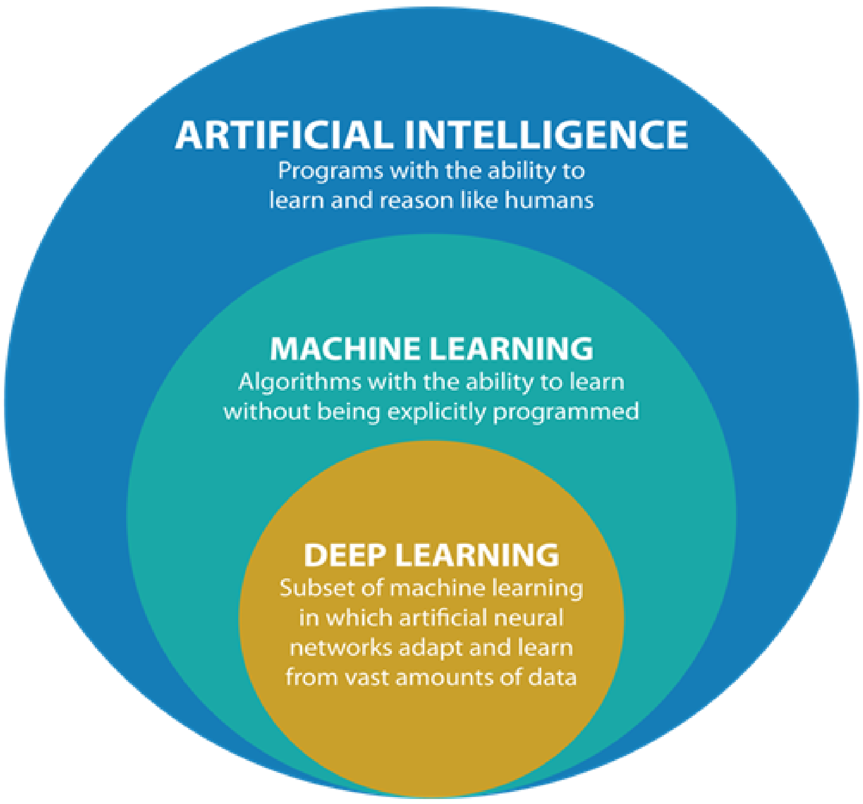

ARTIFICIAL INTELLIGENCE
Programs with the ability to learn and reason like humans
MACHINE LEARNING
Algorithms with the ability to learn without being explicitly programmed
DEEP LEARNING
Subset of machine learning in which artificial neural networks adapt and learn from vast amounts of data

Figure 1.6: Venn Diagram of AI

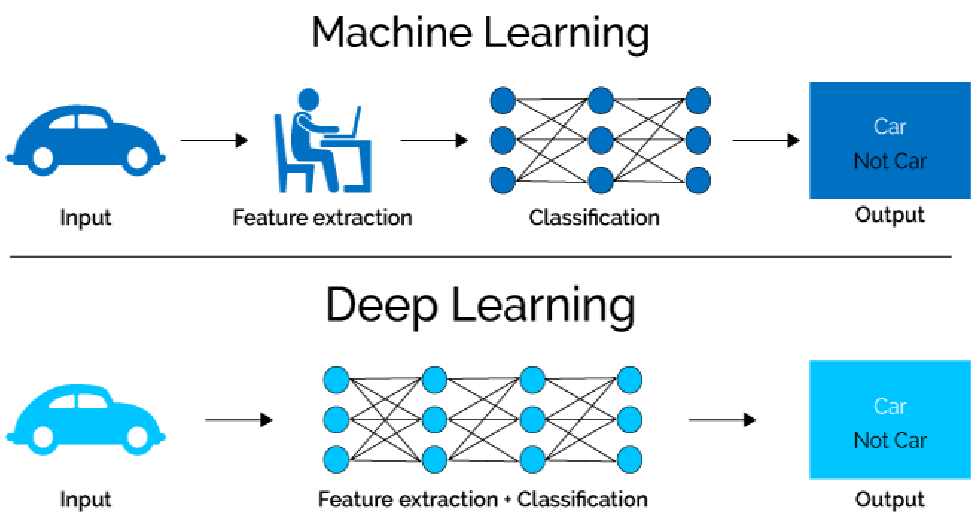


Figure 1.7: Difference between ML and DL

3 it is termed ‘deep neural network’(DNN). Computer vision application involves DNN composed of more than 50 layers.

Perceptron

The most basic unit of neural network is known as the perceptron. An invention of Mr. Frank Rosenblatt as a binary classifier which is composed mainly of three components.

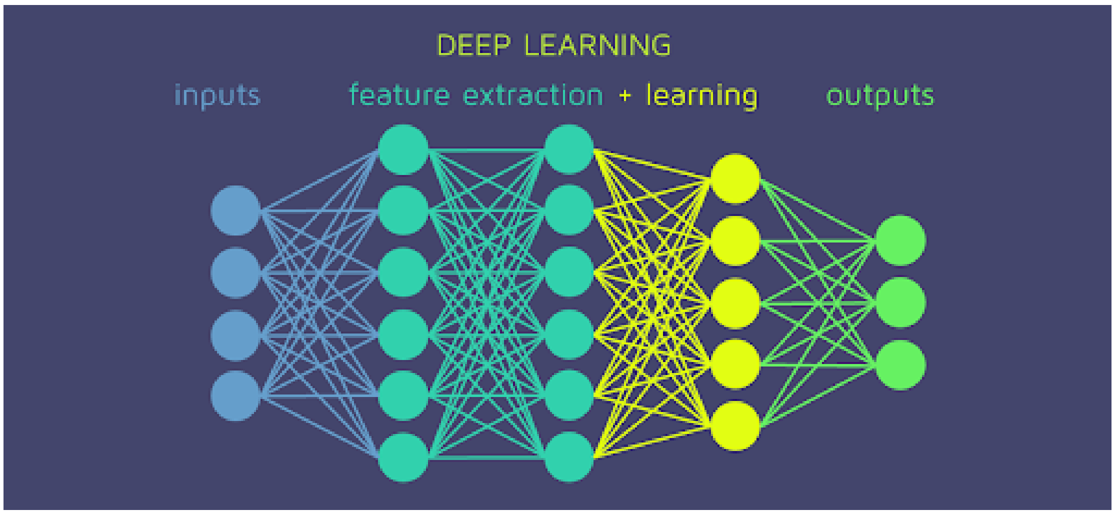


Figure 1.8: Deep Learning

These are as follows:

- Input nodes
- Weight and biases
- Activation function

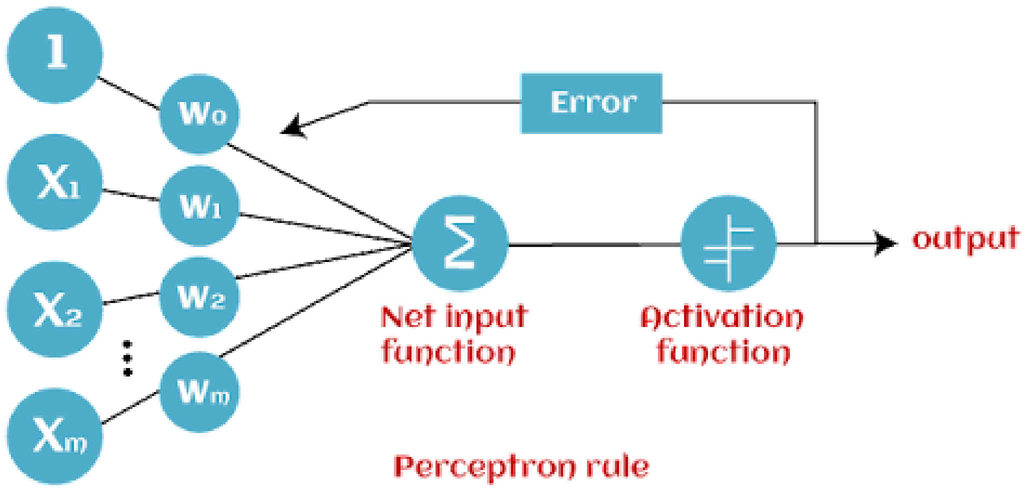
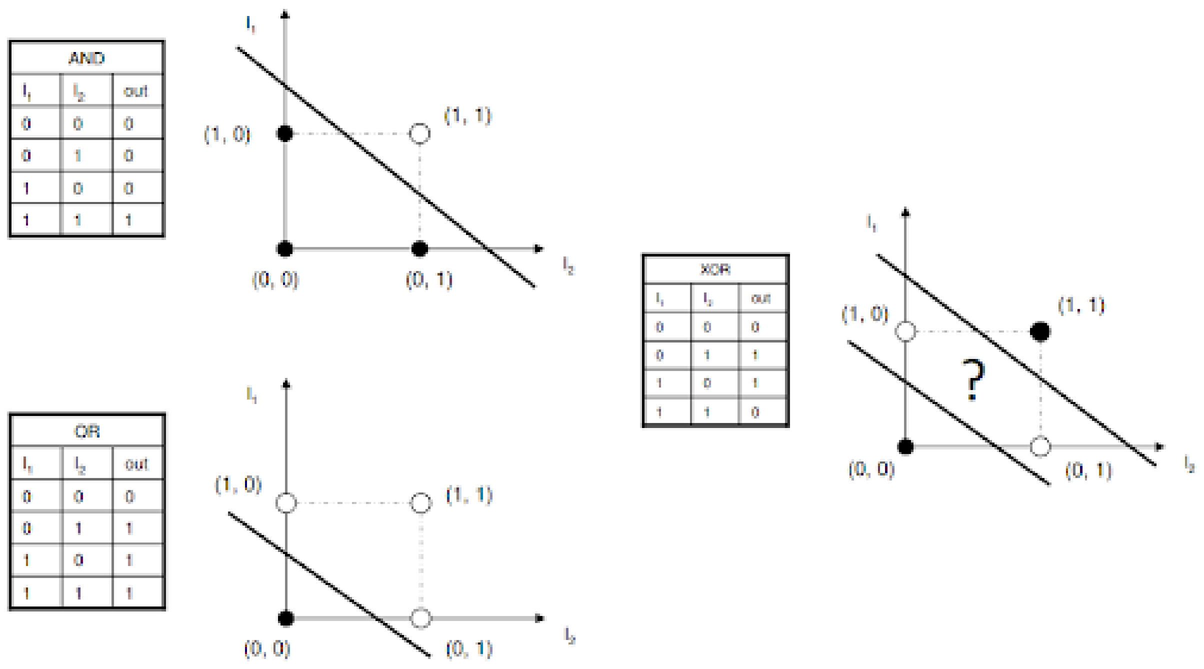


Figure 1.9: Perceptron

Perceptron uses a nonlinear function, here we call it activation function such as sigmoid, tanh, step function, etc.
Working is like, every input value is added at the summation node after multiplying to specific weights, one bais value is also added at the same node. Then these weighted sum values are passed through the activation function to produce output. One perceptron is capable of making only linear decision boundary, hence can make binary

classification, if the decision boundary is a straight line.
As we can simulate AND , OR , NOT gate but can not simulate XOR and XNOR through single perceptron as its decision boundary is nonlinear.



AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1

OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0

Figure 1.10: AND, OR and XOR Gate

This problem can be solved when we use multiple perceptrons at work. That is known as multi layer perceptron (mlp).

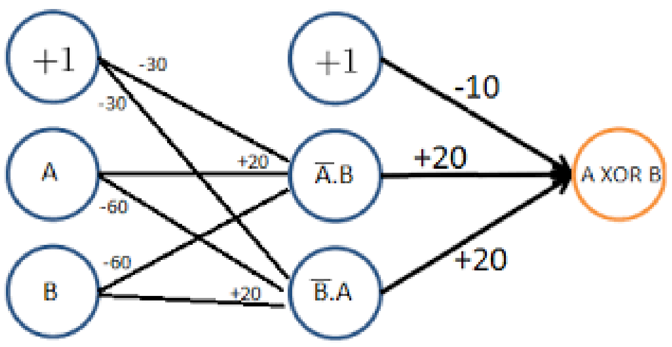


Figure 1.11: Perceptron performing XOR operation

In Fig., three perceptrons are connected to simulate XOR gate.

Feed Forward Neural Network

When connections of perceptrons are built in forward direction and information is only allowed to be processed in one direction it is termed as feedforward neural network. It is the simplest type of neural network used in deep learning.

Working of Neural Network

Deep learning is also classified as supervised learning with certain exceptions. It requires labeled data for training. For images labeling is also done by annotation and segmentation process.

Hyperparameters:

Model architecture

- Number of input neurons (alias perceptron) at input and output layer. This value depends upon the dimension of data we feed to network and output requirements.
- Number of hidden layers. It depends upon the complexity of task.
- Number of Neurons in each hidden layer.

Model compilation

- Cost Function
- Optimizer
- Learning rate
- Number of epoch
- Batch size
- Metric, etc.

After model architecture is built several hyperparameters are set to compile the model. Initially, all weights and biases known as parameters of neural networks are randomly initialized, then the training data is fed at the input nodes of the network. Forward propagation happen through hidden layers which generates output value at output nodes.

Cost/error value is calculated by comparing predicted value of network to the actual labeled value, then an optimizer function such as stochastic gradient descent(SGD) or Adam works to reduce the cost function in the right direction.

While training the network over a large set of data the cost value gradually starts decreasing and the model learns to predict or approximate the actual value within tolerable error limits

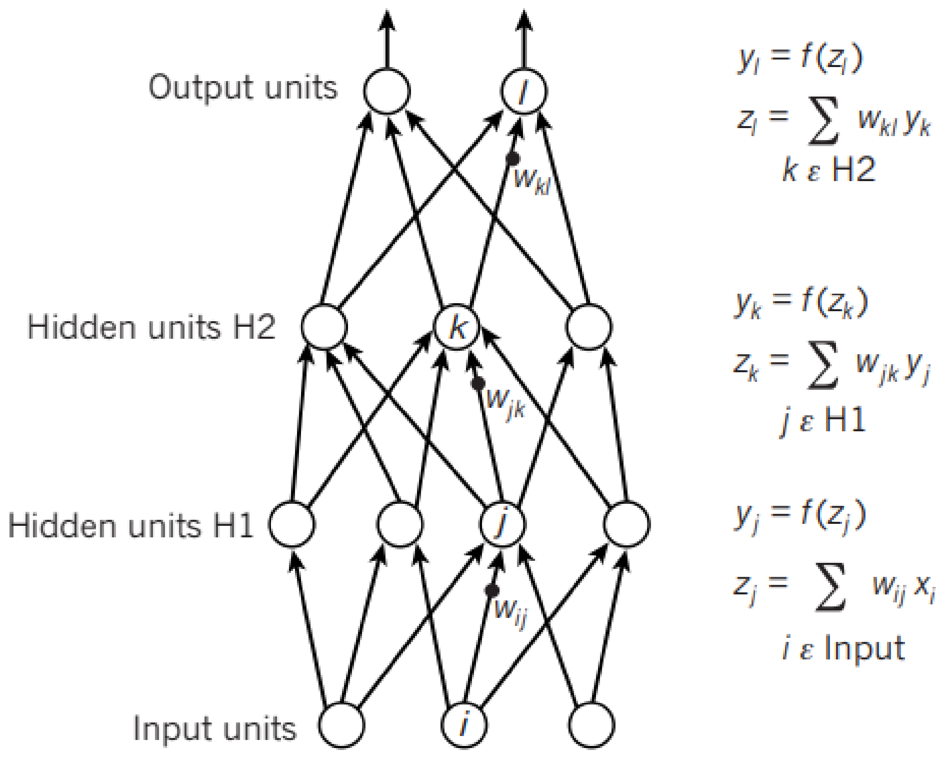

Output units
Hidden units H2
Hidden units H1
Input units
l
k
j
i
w_{kl}
w_{jk}
w_{ij}
$y_l = f(z_l)$
$z_l = \sum_{k \,\varepsilon\, H2} w_{kl} y_k$
$y_k = f(z_k)$
$z_k = \sum_{j \,\varepsilon\, H1} w_{jk} y_j$
$y_j = f(z_j)$
$z_j = \sum_{i \,\varepsilon\, Input} w_{ij} x_i$

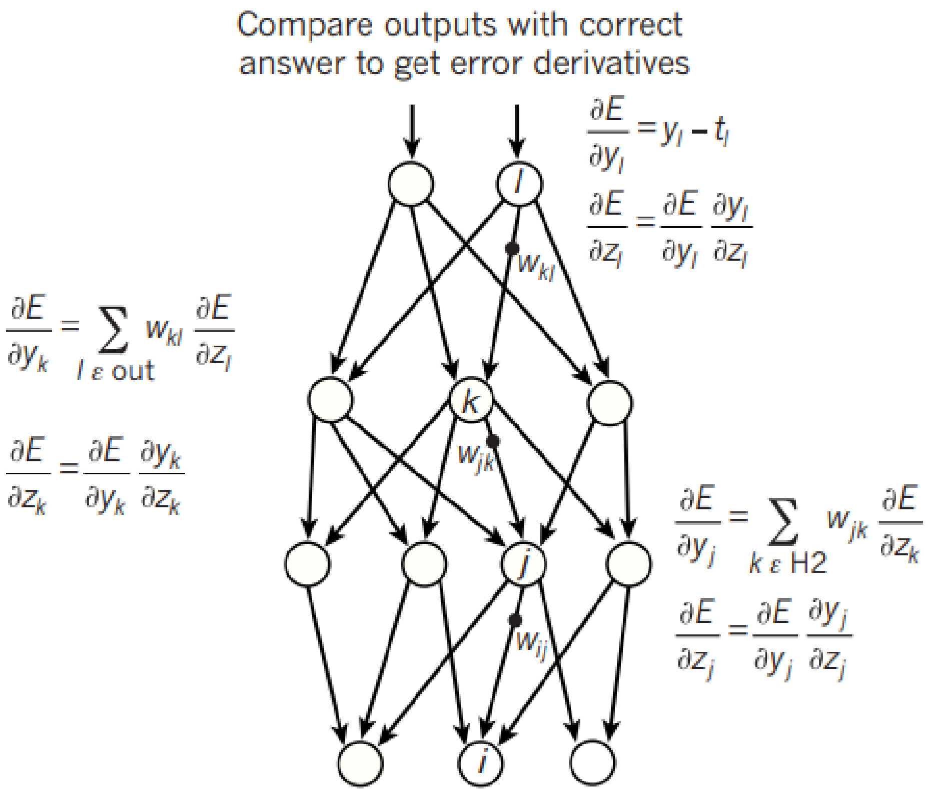
Figure 1.12: Forward propogation of Neural network

Backpropogation: Process of learning involves backpropagation, when a cost/error value calculated after each batch of training, along with optimizer to reduce the subsequent training error weights and biases are changed accordingly with help of partial derivative and chain rule.

Deep learning is now a vast field of research in itself. Different model architecture designed and trained by bigger institutions are sometimes used as standardised model in various applications.

GANs, variational autoencoder, Belief Network are some examples of different techniques that exploit neural network in other way. Transfer learning is another way to make the process of training faster as it uses previously trained model on another dataset.

Convolutional neural network, recurrent neural network are some modified forward neural network which helps in addressing the challenges of computer vision (CV) and natural language processing (NLP) fields.



Compare outputs with correct answer to get error derivatives
$\frac{\partial E}{\partial y_l} = y_l - t_l$
$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l} \frac{\partial y_l}{\partial z_l}$
$\frac{\partial E}{\partial y_k} = \sum_{l \varepsilon \text{out}} w_{kl} \frac{\partial E}{\partial z_l}$
$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_k}$
$\frac{\partial E}{\partial y_j} = \sum_{k \varepsilon \text{H2}} w_{jk} \frac{\partial E}{\partial z_k}$
$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_j}$
l
w_{kl}
k
w_{jk}
j
w_{ij}
i

Figure 1.13: Backpropogation of Neural network

1.1.4 Reinforcement Learning

"This was simply the idea of a learning system that wants something, that adapts its behavior in order to maximize a special signal from its environment."

Sutton Barto 1979

Reinforcement learning (RL) is the idea of learning by taking an action in an environment and waiting for results to continue or to change the action in hope to get better results. It is different from supervised learning where a labeled data is used to guide the model to the right track, here a reward signal is generated in a feedback loop to build a policy for taking decisions on future actions.

Several dynamic problems are there whose labeled data cannot be generated or features can't be extracted easily hence it is not possible to use supervised learning algorithms everywhere.

Examples like helicopter stunt maneuvering, winning a game of Go from the world's best player, playing atari games, controlling a power station etc. Reinforcement learning is in the real sense of mimicking how humans learn from experience. The development

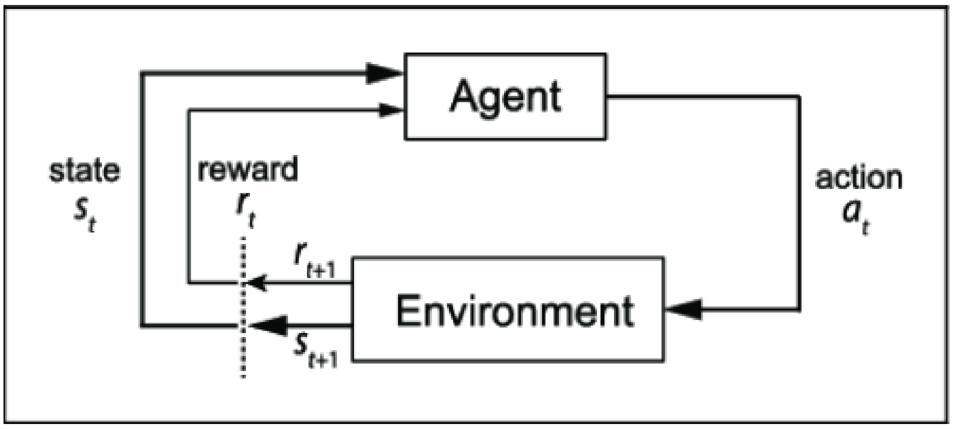


Figure 1.14: Framework of RL

of RL involves many fields of science as shown in fig.

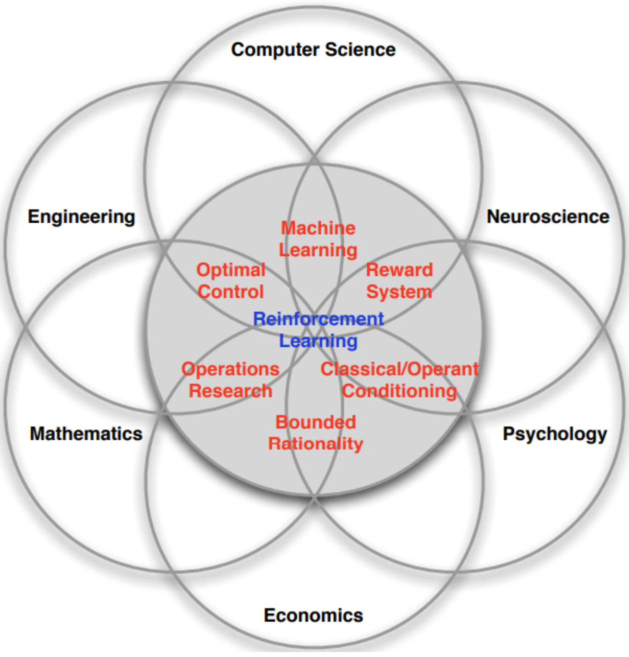


Figure 1.15: Many Faces of RL

RL can work in any sequential decision making scenario. Where the goal could be to maximize the total future reward by selecting appropriate actions. Sometimes it may be better to look for a long term reward by sacrificing the immediate reward.

Taxonomy of Reinforcement learning

- Reward : A reward (R_t) is a feedback signal which indicates how well an agent is performing at any time step t. Typically it is a scalar value. By reward hypothesis it can be claimed that all goals or end learning can be described by maximizing the expected cumulative reward.

- Agent : An agent executes action (A_t) at each time step t and Receives observation (O_t) and a scalar reward (R_t) from environment.
- Environment: Environment accepts the action At from agent in response emits the observation (O_{t+1}) and scalar reward (R_{t+1}). Where, t increments at every environment step.
- History and State: The history is a sequence of observation from the environment, action from the agent and rewards in feedback to the agent from the environment.

  $H_t = O_1, R_1, A_1, ..., A_{t-1}, O_t$

The agents select action in view of history. State is the information which is function of history which determines what happens next $S_t = f(H_t)$

Components of RL agent An RL agent may include one or more of these components:Policy, Value function, and Model.

- Policy: A policy is an agent's behaviour which map it's action from state. It can be deterministic or stochastic. Policy is denoted by π.

  Deterministic policy: a = $\pi(s)$
  Stochastic policy: $\pi(a|s) = P[A_t = a|S_t = s]$
- Value Function: Value Function of a state is a prediction of future reward which is used to evaluate the state's goodness or badness. Considering the value function of a state agent, select between the actions.

  $v_\pi(s) = E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s]$

  Where, γ represents the discount factor. $\gamma \in [0,1]$
- Model: Model is the agent's understanding of the environment, which predicts what the environment will do next and what would be the reward.

To understand above terminology we see an example of maze problem as shown in Fig.1.15 and steps of the solution using RL are as follows:

- Fig1.15(a) shows a maze, where the aim of the agent is to reach from start to goal from the shortest route. This environment can be described by each block as a state s.
- Action space : Moving one step north, south, east, and west.

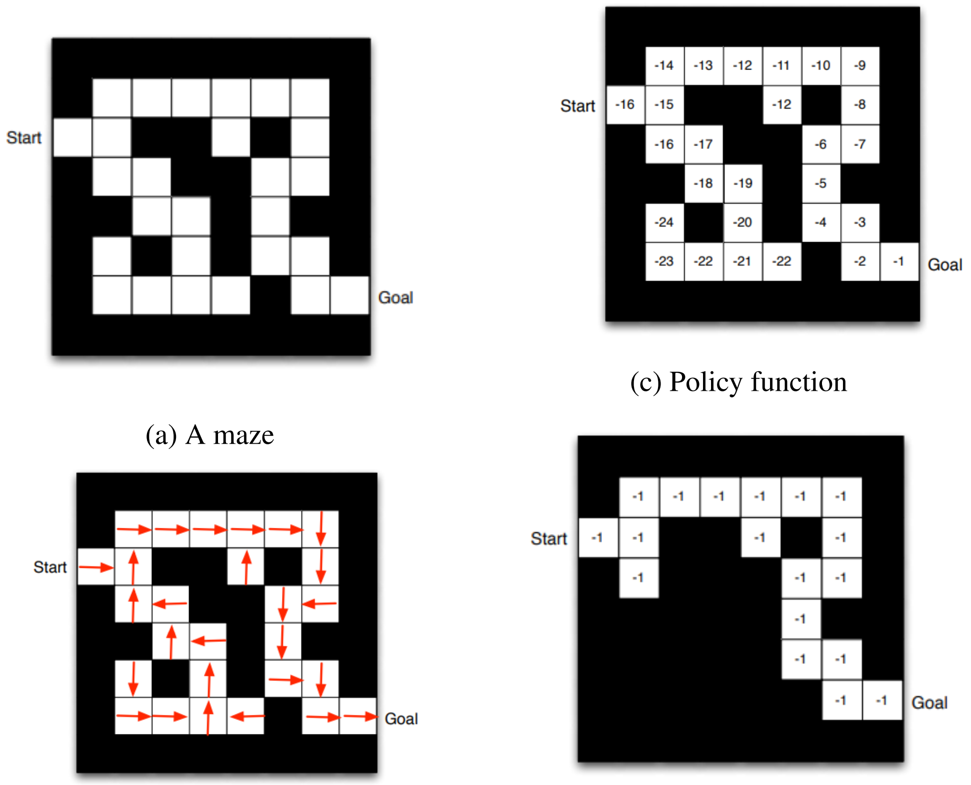
  After interacting with the environment, agents have created a policy to solve this maze problem. Here red arrows show the action to take in that particular state. Hence arrows represent policy $\pi(s)$ for each state s.

- After interacting with the environment, agents have created a value function to solve this maze problem.

  Value representing how good that state is in terms of getting reward and hence movement of agent should be direction of higher.

- After interacting with the environment, agents have created a model approximation to solve this maze problem.

  Here, how actions change the state to reach a goal from start via the shortest route is learned. Agent perception of Environment may be imperfect.



(a) A maze

(b) Value function

(c) Policy function

(d) Model-free

Figure 1.16: A maze problem solved using RL.

Based on all above concept , RL can be classified in various ways as shown in Fig.1.16.

Exploration and Exploitation: To be able to learn from experience one must keep a balance between exploration and exploitation. Without exploration one may miss the opportunity to accumulate more reward at the same time exploitation of those learning should be done to get the benefits. In a similar manner RL also follows the exploration and exploitation policies.

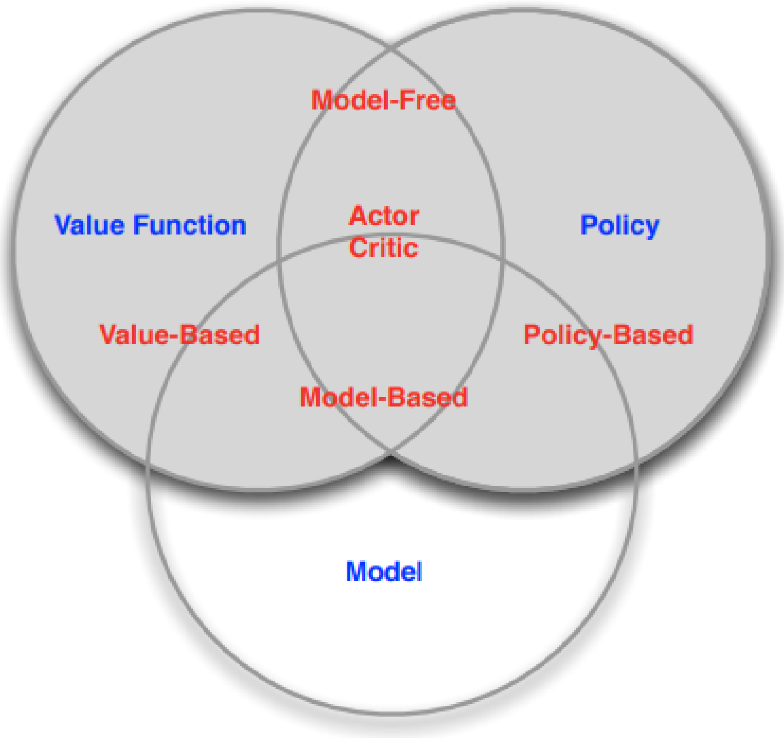


Figure 1.17: Classification of RL

Experience : An experience E which is a tuple of state, action , reward and gamma is stored for each interaction of agent to the environment, which used to build a value function or policy.

$$E=[s_t, a_t, r_t, \gamma]$$

Where s_t, $a_t \in A$, and r_t are the state, action and reward at each time step, respectively.

As we have seen, the RL is the quest of finding the optimal policy for an agent by learning from experience in a given environment guided by rewards and punishments. In this section we see how a value function leads to the optimal policy.

So an optimal Policy π^* must exist that is better than or equal to all other policies, $\pi^* \geq \pi, \forall \pi$

So as a policy is nothing but probability distribution over agent actions given the environment states, where the value function of a state assigned by agent, selects between the action.

$$\pi(a|s) = P [A_t = a \mid S_t = s]$$

So another function named as state-value function $v_\pi(s)$ is defined as the expected return gained when an agent start from state s, and then follows the policy π .

$$v_\pi(s) = E\pi [G_t \mid S_t = s]$$

Hence, the optimal state-value function $v^{*}(s)$ can be defined as the maximum value function of state assigned by an agent following all the other policies.

$$v^{*}(s) = max\ \pi\ v_{\pi}(s)$$

Similarly, the optimal action-value function $q^{*}(s,\ a)$ can be the maximum action-value function over all policies.

$$q^{*}(s,\ a) = max\ \pi\ q_{\pi}(s,\ a)$$

So an optimal policy can be found by maximizing over $q{*}(s,\ a)$,

$$\pi * (a|s) = 1,\ \text{if a = argmax}\ a{\in}A\ q^{*}(s,\ a)$$
$$0,\ \text{otherwise}$$

RL Algorithms

Formulation of the RL problem is done in the context of the markov decision process where, each state is a markov state, which means the state has all the relevant information from history. State and Action must be a finite set.

The markov decision process is solved by iterative methods to find optimal policy. Some algorithms given below:

- Value Iteration
- Policy Iteration
- Q-learning
- SARSA

1.1.5 Deep Reinforcement Learning

As neural networks are the best approximator, we infuse neural networks with RL to obtain the optimal policy. This combination of DL and RL makes a significant step towards the general AI. Where, an agent can learn itself and make decisions based on interaction with the environment. As DRL [mnih2013playing] explores through the environment it swaps through all state-action pairs and builds the optimal policy. Hence a simulation environment is needed for the initial phase of learning to avoid the damage of physical resources as learning can be unstable at certain times. Later the trained agent from the simulation can be used in the physical world. So now the hurdle in realization of general AI is the realization of simulation software as much close possible to the real world.

There are several problems with the iterative method of solving markov decision process like non convergence, high computational time etc. so instead we use deep learning methods to approximate the values like action- value function and state-action function in the RL framework which results in better learning. Hence this field is known as Deep Reinforcement learning. Various algorithms have been developed in DRL over time and tested on different games. Some algorithms names are DQN, DDQN, A2C/A3C, PPO etc. we have used DDQN and A2C in our project which will be explained in the section proposed approach.

1.2 Introduction to Optical Networks and Quantum Key Distribution

Optical communication networks carry about 99% of the global Internet traffic [6]. In order to support the exponentially increasing bandwidth due to high-speed broadband, Internet of Things, and datacenter interconnection in the beyond 5G era, research efforts are being made to design and develop high-capacity optical communication networks. Meanwhile, it has been observed that the data transferred through an optical network has suffered from increased eavesdropping and interception, for example, in 2015, there were 16 known attacks on fiber optic cables just in the San Francisco area [7]. Thus, security of optical network against such types of attacks is very important [8]. Therefore, conventional data encryption has been adopted to prevent optical communication networks against such security attacks [9]. However, the security of such encryption technique relies on the platform of computational complexity, and the evolution of quantum computers in the near future will easily break the security of such systems [10]. Hence, this makes optical communications networks insecure.

One of the most promising solution to prevent optical communication network is quantum key distribution (QKD), relies on the fundamental principles of quantum physics [11]. The fundamental principles of quantum mechanics, i.e., quantum no-cloning theorem and the Heisenberg uncertainty principle, used to prove that two remote end-nodes of an optical link can generate a shared random secret key known only to them by using specific QKD protocols, such as BB84.

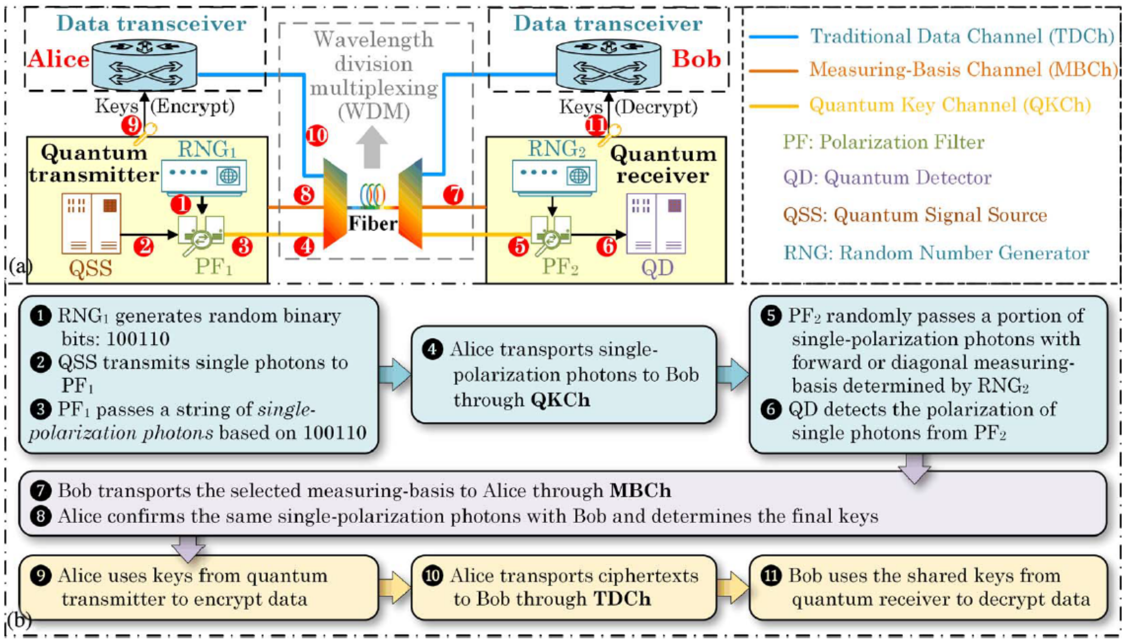
The integration of QKD with the existing optical communication networks has been proposed as a cost-efficient solution, however, this integration introduces various networking challenges. In order to establish connection between two end-nodes in the QKD networks, apart from data communication channel two additional channels, i.e., quantum channel and classical channel are required. Hence, the problem of routing and resource allocation for three types of channels in such networks is one of the most challenging issues to be addressed [8, 9]. The complete process of data encryption and decrytion using QKD is shown in Fig..

A basic point-to-point mechanism of QKD system is shown in Fig. 12, as described in [8]. In QKD system, Alice's lab consists of a quantum transmitter, which consists of quantum signal source (QSS), random number generator (RNG), and polarization filter (PF) and Bob's lab consists of a quantum receiver, which includes quantum detector (QD), RNG, and PF [8]. The steps involved in establishing secure connection between Alice and Bob in Fig. 12 are described below [12]:

- In the Alice's lab, the QSS transmits single photons to the PF; and RNG generates random bits and sends them to the PF.
- The single photons are polarized with one of the four polarization states (H, V, 45°, 135°). The bits generated by RNG are encoded with the polarized single photons to obtain qubits.
- Alice sends the qubits to Bob through quantum channel, and classical channel is required for qubit synchronization between Alice and Bob.
- In Bob's lab, the quantum receiver receives and measures the qubits with randomly selected polarization bases.
- Alice and Bob exchange the measuring bases with each other via classical channel and compare them. Later, The qubits with the same polarization bases are considered for secret key generation. The sequence of bits obtained after the comparison of bases constitutes the sifted key.
- Alice and Bob may not be sure about the correctness of the bits considered for the sifted key. Thus, to further ensure the correctness , error correction, privacy amplification, and authentication are performed . The remaining bits obtained after these processes (referred to as post-processing) constitute the secret key. Alice uses the generated secret key to encrypt the classical data and transmits the encrypted data to Bob via data channel. Bob uses the same key to decrypt the original data.

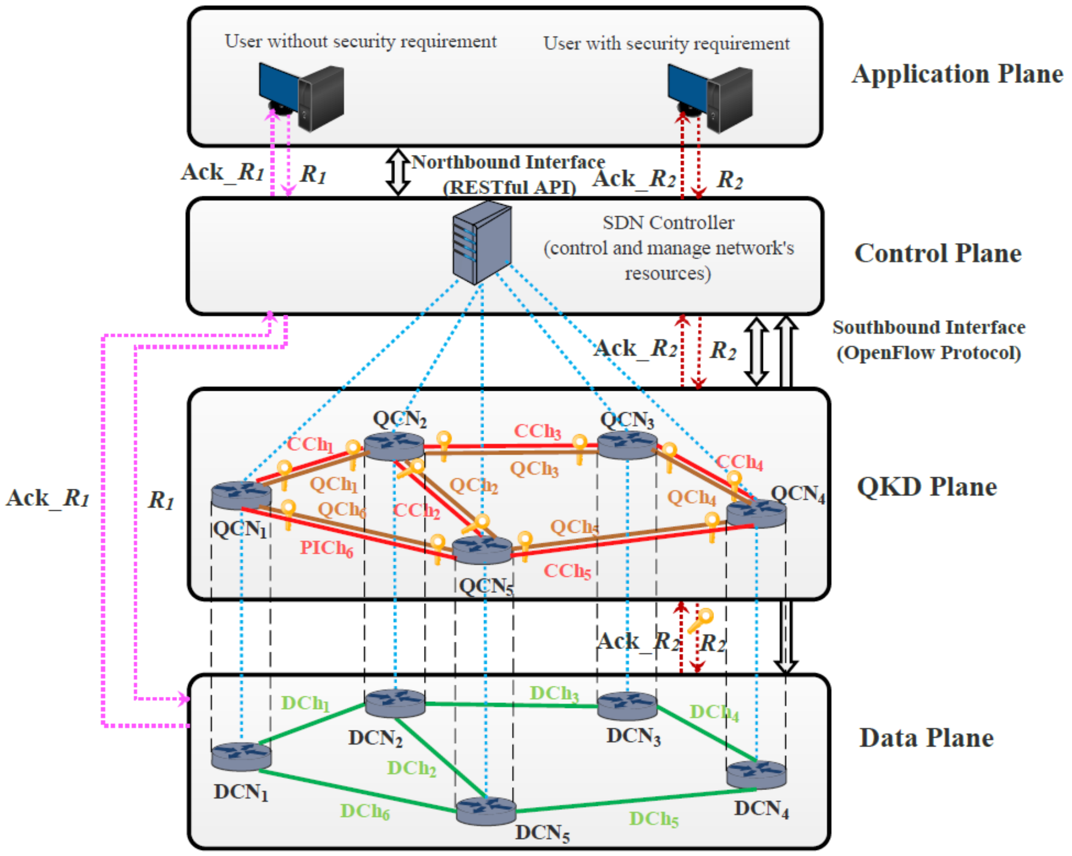
The basic architecture of QKD network is shown in Fig. and described below.This architecture consists of four different planes, namely, application plane, control plane, QKD plane, and data plane [7, 12].

- In the application plane, requests are generated and then transferred to the control plane for further processing. The status of request acceptance or rejection is received at the application plane.



Data transceiver
Alice
Keys
(Encrypt)
Quantum transmitter
RNG₁
QSS
PF₁
Wavelength division multiplexing (WDM)
Fiber
Data transceiver
Bob
Keys
(Decrypt)
Quantum receiver
RNG₂
PF₂
QD
Traditional Data Channel (TDCh)
Measuring-Basis Channel (MBCh)
Quantum Key Channel (QKCh)
PF: Polarization Filter
QD: Quantum Detector
QSS: Quantum Signal Source
RNG: Random Number Generator
(a)
1 RNG₁ generates random binary bits: 100110
2 QSS transmits single photons to PF₁
3 PF₁ passes a string of single-polarization photons based on 100110
4 Alice transports single-polarization photons to Bob through QKCh
5 PF₂ randomly passes a portion of single-polarization photons with forward or diagonal measuring-basis determined by RNG₂
6 QD detects the polarization of single photons from PF₂
7 Bob transports the selected measuring-basis to Alice through MBCh
8 Alice confirms the same single-polarization photons with Bob and determines the final keys
9 Alice uses keys from quantum transmitter to encrypt data
10 Alice transports ciphertexts to Bob through TDCh
11 Bob uses the shared keys from quantum receiver to decrypt data
(b)

Figure 1.18: Point-to-point QKD mechanism



User without security requirement
User with security requirement
Application Plane
Ack_R₁
R₁
Northbound Interface (RESTful API)
Ack_R₂
R₂
SDN Controller (control and manage network's resources)
Control Plane
Southbound Interface (OpenFlow Protocol)
QCN₁
QCN₂
QCN₃
QCN₄
QCN₅
CCh₁
CCh₂
CCh₃
CCh₄
CCh₅
QCh₁
QCh₂
QCh₃
QCh₄
QCh₅
QCh₆
PICh₆
QKD Plane
DCN₁
DCN₂
DCN₃
DCN₄
DCN₅
DCh₁
DCh₂
DCh₃
DCh₄
DCh₅
DCh₆
Data Plane

Figure 1.19: Basic network architecture

- The control plane consists of the software-defined networking (SDN) controller that controls and manages the resources of network. The control plane allocates resources to requests from the quantum channel (QCh), and data channel (DCh) in the QKD plane, and data plane, respectively.

- The QKD plane consists of quantum communication nodes (QCNs) and the connection among QCNs is established over quantum channel and classical channel (CCh). The implementation of QKD plane is dependent on the specific QKD protocols. The process of secret key generation between each nodes in the network takes place in the QKD plane.
- Data plane: The data to be transmitted over data channel is encrypted using the conventional encryption techniques, by the secret keys generated at the QKD plane.

1.3 Organization of Thesis

The subject matter of the thesis is presented in the following five chapters,

- Chapter-1: This chapter gives an overview of the Artificial Intelligence (AI) which includes levels of AI, introduction to machine learning (ML), deep learning (DL) and neural network (NN), reinforcement learning (RL), and deep reinforcement learning (DRL). In this chapter, optical networks and quantum key distribution are also discussed.
- Chapter-2 This chapter includes the literature review of past works and research done using machine learning in various applications. It also also emphasizes the motivation of this research and objectives.
- Chapter 3: This chapter provides a detailed description of our proposed approach along with different DRL models used in this work.
- Chapter-4: This chapter discusses the results obtained from the proposed algorithm compared with existing schemes.
- Chapter-5: This chapter highlights the new findings obtained by the utilization of ML technique and conclusions arising out of this complete study are elucidated. The scope for future and continuation of this research work are also reported.

Chapter 2

Review of Past Work and Problem Formulation

This chapter includes a literature review of the ML and past work done in the field of optical networks using QKD. Section 2.2 includes the motivation behind the work done.

2.1 Related Work

ML and it's derivatives field are largely used in engineering and development of new era technology. Every research field is exploring the use of ML In there domain. Many got successful and some reach the production level of their product. So let's have a look how different ML field are expanding their horizons in this era.

Machine learning started to be used in computer gaming by Arthur Samuel in 1959, a pioneer in AI , then advances came along time[]. Nilsson wrote a book about pattern classification and machine learning in 1960. Beginning in 1981, a neural network on a computer terminal began to learn 40 characters. Hinton, Nielsen, Rumelhart, and Williams-Hetch, neural network scientists, demonstrated the multilayer perceptron (MLP) with practical backpropagation (BP) training in 1985 and 1986. The next generation of neural network works began in 2005, with researchers Andrew Ng, Hinton, Bengio, LeCun, and others who provided extensive use of deep learning in the field of computer vision.

ML algorithms still in vast use where data driven decision making is done. Generation of data through sensors in machines or by manually created forms and spreadsheets became common in this era. This data provides new insight to businesses to develop new products and review old ones. Field of data science has extensive use of Ml techniques.

Computer vision, a field where mainly data is in the form of visuals like images and videos, are used to perform many tasks from object detection, classification, and detection of disease to drone warfare and processing satellite images. Deep learning and its advanced method like generative adversarial network are the heart of the computer vision domain. In today's time cameras are everywhere and generation of visual data is easy and common. CCTV cameras are used along with different Ml/Dl algorithms to sense emotion and face recognition to track and catch criminals by government organizations.

Natural language processing (NLP) is a field of computer science where machines are used to understand the complex human language and help to solve multiple applications like chat bot, translation, text to speech etc. This is one of the hot fields of AI where extensive research is going on. This field is powered by a recurrent neural network (RNN) which is a slight modification of neural network by storing the history of inputs used, helping in understanding the context and tenses of long texts.

Reinforcement learning is mostly used in research work and this technique is still under development. RL applications involve those fields where defining a problem with feature space is not sufficient. Like playing games which required decision making and understanding of context. RL algorithm like dynamic programming, SARSA, Q learning were used initially in basic games like Atari video games and several board games. Later deep learning infused with the RL to bring the magic and it termed as DRL.

DRL is a powerful machine learning technique which uses reward based learning and feature extraction of deep learning. DRL is used to make Robots walk on rough tracks and terrain areas whose application would be in rescue or bomb diffusion operation, DRL is getting used in building agent for trading in financial market, self driving cars, in NLP also like text summarization, question answering more of taking decisions as human does. It's capabilities to take decisions, makes it application endless. The only hindrance is the training simulator for that application to train the agent and bring the trained agent in the real scenario. Recently, DRL has been widely used as an emerging technique to address various problems and networking challenges of communication networks.

Several strategies have been proposed in the literature to address the routing and resource allocation problem of QKD networks [8, 9]. For enhancing the security level of request, a concept of secret key updation was introduced and different security level provisioning solutions were proposed in [8, 9]. The secret key of each request must be updated periodically during the key updation process, thereby increasing the difficulty of an eavesdropper to detect the information of the generated key. For maintaining a balance between resource utilization and security level of QKD networks a key on demand scheme for such networks was designed with quantum key pool technique was developed . Furthermore, a time scheduled scheme was designed with quantum key pool technique, for adequate provisioning of secret keys [13]. Moreover, to address the different networking challenges of such networks, i.e., trusted repeater node placement, survivability [14], and QKD for multicast services, of QKD-secured optical networks. However, to address the networking challenges of such network DRL has not been explored much.

2.2 Motivation

DRL has capability to understand the dynamic functionality of environment and make a decision to take actions. Such potential of DRL can be used to address various decision making problems in the field of QKD networks.

As discussed in Chapter 1, the integration of QKD into the existing optical network introduces one of the most important networking challenges, i.e., routing and resource allocation in such networks. Also, the problem of routing and resource allocation in QKD network is different and complex to solve because of the dynamic arrival and departure of requests as well as the unique concept of key updation for enhancing the security level of request. However, not much work has been done to address this problem in such networks. Therefore, we explore DRL technique to solve such complex decision making problem of QKD network.

Chapter 3

Proposed Approach

In this chapter, we describe the system model used in this work. Also, we explain the concept and the operational principle of the proposed approach along with different DRL model used in this work for training.

3.1 System Model

We consider a physical QKD network topology of an optical network as a directed connected graph *G(N, L, W, w)* where, *N* is set of nodes, *L* is set of links. *W* represents the total number of available wavelengths on each link and *w* represents the total number of reserved wavelengths for quantum channel on each link. The following notations are used in this work with their definition:

- *N* and *L* are the total number of nodes and links, respectively, in the network.
- *W* is total number of wavelengths on each fiber link.
- *w* is total number of wavelengths reserved for quantum channel.
- *Req* is set of incoming requests.
- $req(s_{req}, d_{req}, T, t_i, t_u)$ is a request, where $req \in Req$.
- s_{req} and d_{req} are the source and destination nodes of a request.
- *T* is the update period of a request.
- t_i is the required number of time-slots for initial key of a request.
- t_u is the required number of time-slots for updated keys of a request.
- *K* is the number of pre-calculated routes of a request.
- *k* is a route of a request, $k \in K$

3.2 Proposed Approach

3.2.1 Concept and Operation Principle of DRL-based resource allocation

The proposed DRL-based resource allocation (DRA) approach solve the problem of routing and resource allocation of QKD network using DRL technique. The objective of the DRA approach is to maximize the success probability of requests in the QKD network during allocation and re-allocation.

In the DRA, the agent selects a route from the pre-calculated K routes, where $k \in K$ and allocate network resources on the selected route using the first-fit (FF) resource allocation scheme. The FF [40] resource allocation policy indexes all the time slots and maintaining a list of used and unused time slots. From the list of available time slots, this policy selects the lowermost available time slot and allocates it to serve the request shown in Fig. 3.1. When a request is completed, the used time slots are released and added to the list of available time slots. The FF resource allocation policy has been utilized in this work because of its low computational complexity and simplicity.

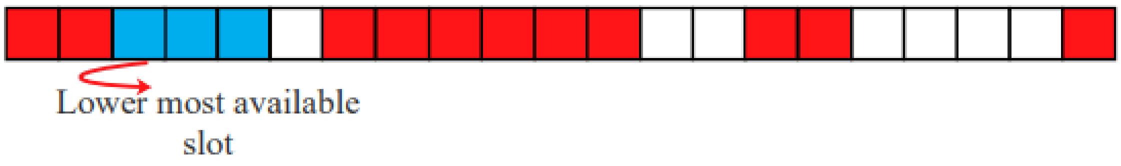


Figure 3.1: First-fit

We consider a network topology with 5 nodes and 7 links for explaining the operation principle of DRA approach as shown in Fig. 3.2. Assume different requests, i.e., AB, AC, and BC generates in the network. The routes of these requests are shown in Table 3.1.

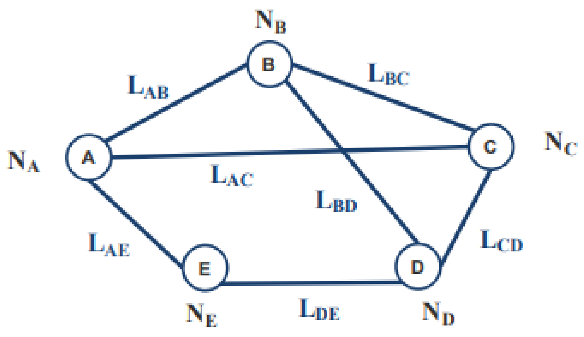


Figure 3.2: Network Topology

Table 3.1: Routes of Request

Request	**Route 1**	**Route 2**	**Route 3**	**Route 4**
AB	A-B	A-C-B	A-E-D-B	A-E-D-C-B
AC	A-C	A-B-C	A-E-D-C	A-E-D-B-C
AD	A-B-D	A-C-D	A-E-D	A-B-C-D
AE	A-E	A-B-D-E	A-C-D-E	A-B-C-D-E
BC	B-C	B-A-C	B-D-C	B-A-E-D-C
BD	B-D	B-C-D	B-A-C-D	B-A-E-D
BE	B-A-E	B-D-E	B-C-D-E	B-C-A-E
CD	C-D	C-B-D	C-A-E-D	C-B-A-E-D
CE	C-A-E	C-D-E	C-B-A-E	C-B-D-E
DE	D-E	D-B-A-E	D-C-A-E	D-C-B-A-E

Consider a *req AC* arrives in the network and the routes of *req AC*, are shown in Table 3.1. On receiving a *req AC*, the SDN controller retrieve the state representation (includes request and network resources information). The information is processed through DNN and an action a_t, i.e., selection of a route from pre-calculated routes, i.e., *A-C, A-B-C, A-E-D-C, A-E-D-B-C*, is taken according to the trained policy $\pi(A|s_t)$.

Let us suppose, a route *A-B-C* is selected for a *req AC*. Now, AC searches the availability of the resources on each corresponding link, i.e., A-B and B-C for establishment using the FF algorithm. If resources are available during allocation and re-allocation then a *req AC* is accepted, otherwise rejected. While training, the reward of +1 is awarded for the acceptance of request, otherwise reward is -1. The policy $\pi(A|s_t)$ will be modified over training based on the obtained reward in order to achieve the optimal result.

3.2.2 Training

DRL is new in machine learning and extensive research is going on seeing the enormous potential. Many algorithms have been developed and tested on various environments. Training of DRL is a sophisticated process,as one has to look out for many hyperparameter settings and formulation of reward function. Sometimes training becomes unstable and DRL agent get confused . So there are certain famous algorithms on which training is stable and performance in various environments are promising. Hence we have explored two such algorithms in our thesis to achieve the objective.

We designed the training of DRA based on the framework of DQN and A2C algorithm. Simulation model of QKD networks and its operation is built in python and libraries of RL algorihms written in pytorch are used. We used the jupyter notebook for training of the DRL agent.

DQN

The first DRL agent was selected is DQN. DQN stands for Deep Q- network. In this DRL algorithm the Q- learning technique is combined with a deep convolutional neural network model to serve the purpose. In RL, Q-Learning is used to give solutions for the control side of the problem, leaving the estimation side to the temporal difference (TD) learning. Q learning is model free as well as an off-policy learning method. Q-learning typically uses the ε - greedy policy to reach the optimal policy.

In Q-learning, Q stands for quality. Basically Q-learning focuses on quality of state-action value, i.e., *Q(s,a)* instead of approximating the value function of the state, TD method is used for estimation. A Q table maintains the value of *Q(s,a)* pair and defines the estimated optimal policy. In each environment step an action is taken independent of the estimated optimal policy (hence off-policy learning), generally a random action policy is used while exploration , and updation of the *Q(s,a)* pair value in table is done in a greedy manner, i.e., max value of *Q(s,a)* is retained.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \, [R_{t+1} + \gamma \, max_a \, Q(S_{t+1}, a) - Q(S_t, A_t)]$$

After sufficient interaction of the q-learning agent to the environment happens the updation in Q table of *Q(s,a)* value mostly remains unchanged and hence it estimates the optimal policy.

Now for a small environment maintaining the Q table is easy but for an actual real life problem state action pairs are in large numbers given the complexity and dynamic nature of the environment which can not be easily maintained and updated. So to estimate the optimal policy neural network comes in handy which takes care of the estimation and updation of large *Q(s,a)* pair values.

DQN is hence a powerful DRL algorithm which was initially used to play games like alpha GO, atari games, carte pole etc.. It uses a convolution neural network to the features from the frame of the game and estimates an optimal policy through a fully connected feedforward neural network. Softmax activation function are used at the output layers to select the action from action space.

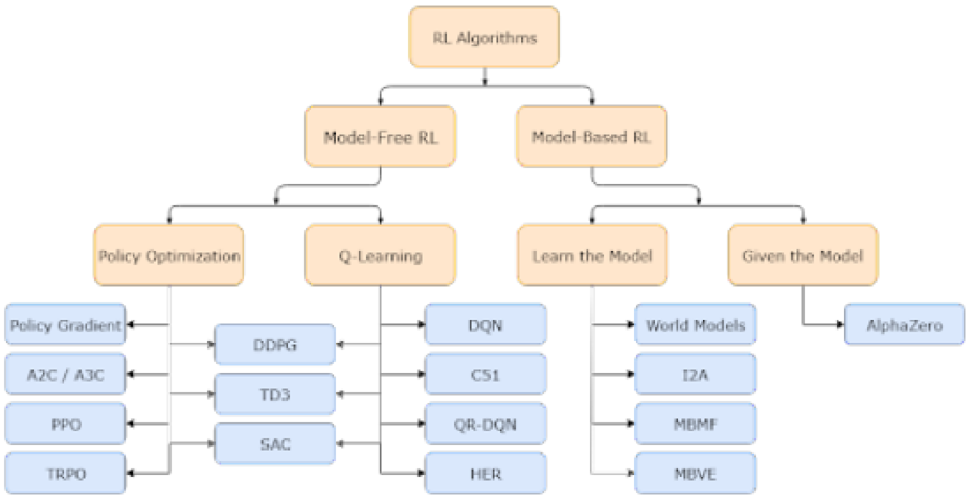


Figure 3.3: DRL Agents

Double DQN employs two neural network models that are identical. One learns during the experience replay, similar to DQN, and the other is a clone of the first model's final episode. This second model is used to calculate the Q-value.as The max operator in normal Q-learning and DQN employs the identical values to pick and evaluate an action. As a result, overestimated values are more likely to be chosen, resulting in overoptimistic value assessments. To avoid this, we can separate the action selection process from the estimate of Q value. Several models are built over time like Double DQN, Duelling DQN whose core is DQN model to overcome the challenges in training the DQN agents.

A2C/A3C

Algorithms like policy gradients suffer from instability and slow convergence. To overcome such problems and make a balance between the algorithm of policy approximation and value approximation , the Advantage Actor critic method has been developed.

The Advantage Actor Critic has two main variants: the Asynchronous Advantage Actor Critic (A3C) and the Advantage Actor Critic (A2C). A3C is asynchronous because it implements parallel training where multiple workers (agents-environment interaction) in parallel environments independently update a global value function.

We need a policy to take any action in reinforcement learning. Because the actor is taking the action in the actor-critic model, this policy must be with the actor. As a result, the actor has a (stochastic) policy that it uses to take action at each stage of training,

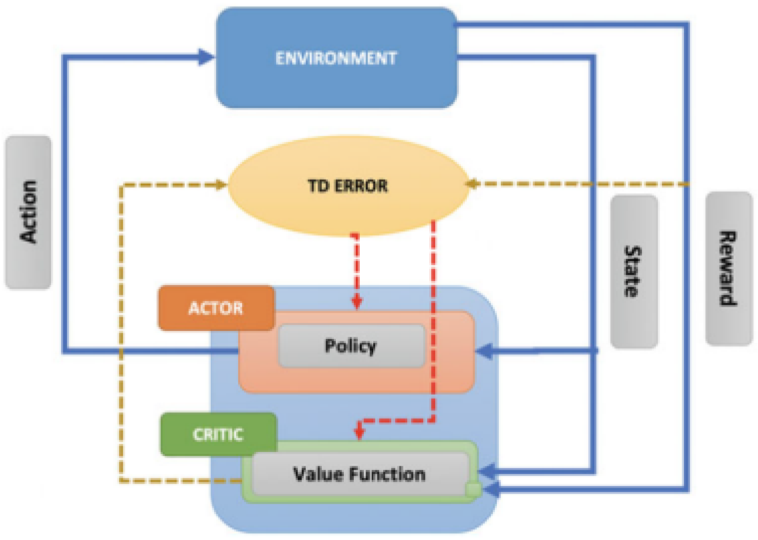


Figure 3.4: High-level architecture of actor-critic method

while also improving its policy (approximation) using the policy-gradient approach, During training, the actor's policy is constantly updated, and this policy is updated using the (state) value estimates received from the critic. The value estimates of the critic serve as a baseline for the actor to update its policy using the policy-gradient approach.

Hence two neural networks are used, One is to evaluate the next action i.e, for actor a policy estimator which will be also used for selecting action and other one to evaluate the value estimate, i.e, Critic. As shown in figure below.

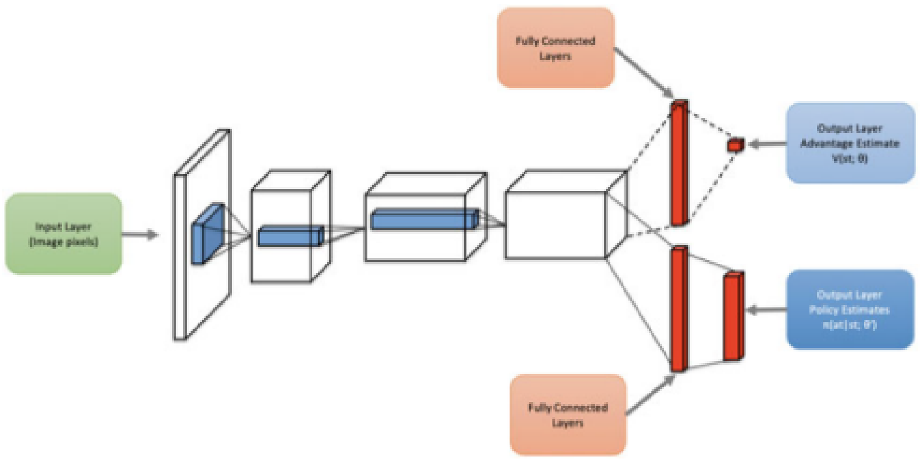

Fully Connected Layers
Output Layer Advantage Estimate V(st; θ)
Input Layer (Image pixels)
Output Layer Policy Estimates π(at|st; θ')
Fully Connected Layers

Figure 3.5: CNN based implementation of Actor- critic

Chapter 4

Results and Discussion

4.1 Simulation Setup

For performance evaluation of the proposed DRA approach, we adopt NSFNET topology (14 nodes and 22 links) for simulations, as shown in Fig. 4.1. We assume a dynamic traffic, where requests are generated using Poisson process following a uniform traffic distribution, with the average arrival rate and service duration being 35 and 20 time units, respectively. between source and destination nodes in the QKD network for each simulation run.

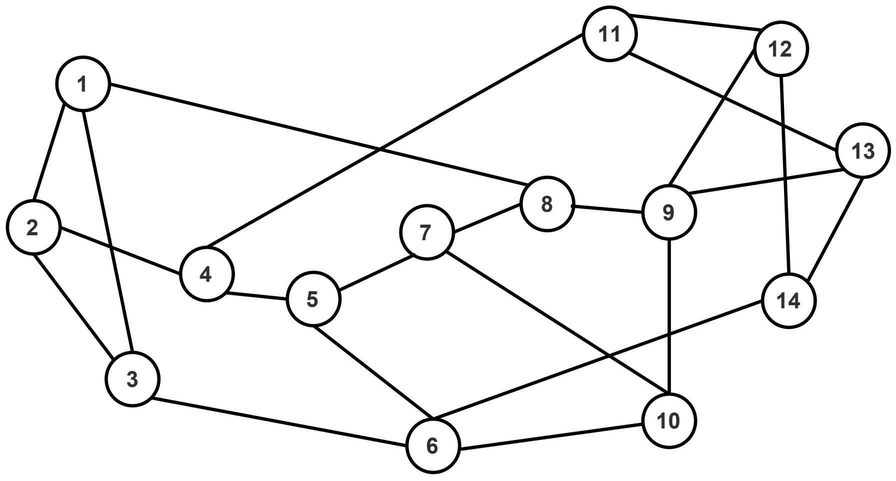


Figure 4.1: NSFNET

Moreover, the sequence of arrival of requests is also varied for each simulation run. The obtained results are averaged over 50 simulations. We consider two metrics, namely, blocking probability (*BP*) and resource utilization (*RU*) of requests to evaluate the performance of DRA with two baseline approaches, i.e., first-fit and random fit (RF). A *BP* is defined as the ratio of the total rejected requests to the total incoming requests in the QKD network. The *RU* is defined as the ratio the resource utilized to the total resource available in the network. We designed the training of DRA based on the framework of DQN and A2C algorithm. Simulation model of QKD networks and its operation is built in python and trained on jupyter notebook. Several open source libraries we have integrated with in our framework that are NetworkX to implement the graph representation of network model and PFRL for RL Algorithms written in pytorch.

4.1.1 Training Results

In this work we have utilized A2C DRL algorithm to train on the QKD simulated environment. Figure 4.1 shows the training result of BP versus training iterations on arrival rate of 0.7. We can observe from the Figure 4.1, as our agent start interacting with the QKD environment, agent's estimated policy converges quickly during initial phase of learning from 0 to 70000. Afterwards, training converges to optimal point which can be seen in Figure 4.1, from 120000 of iterations. The objective of training is to reach an optimal policy by observing the flat graph, which we have achieved around 130000 of iterations.

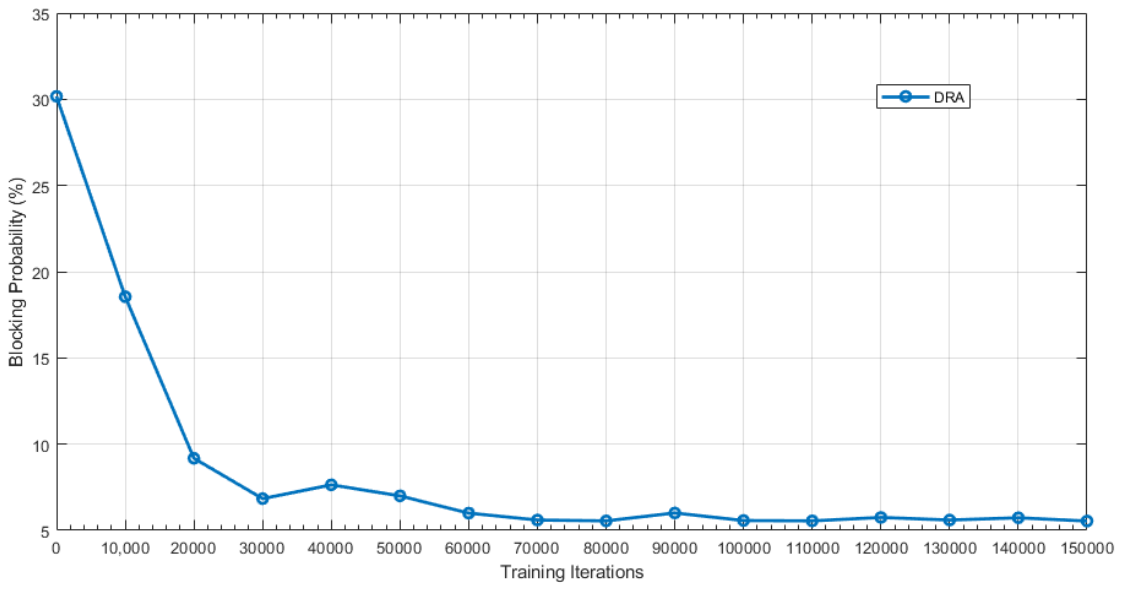


Figure 4.2: Training results of BP versus training iterations

4.1.2 Testing Results

Figure 4.3 and Figure 4.4 show the testing results of BP and RU versus arrival rate. We can observe from Figure 4.3, as arrival rate increases blocking probability also increases in the network due to unavailability of network resources at large arrival rate. It can be seen from plotted graph that the proposed DRA approach performs better as compared to two existing approaches, i.e., FF and RF. Using the proposed DRA approach the BP of requests is reduced by 2.11% and 9.6% in comparison to FF and RF, respectively.

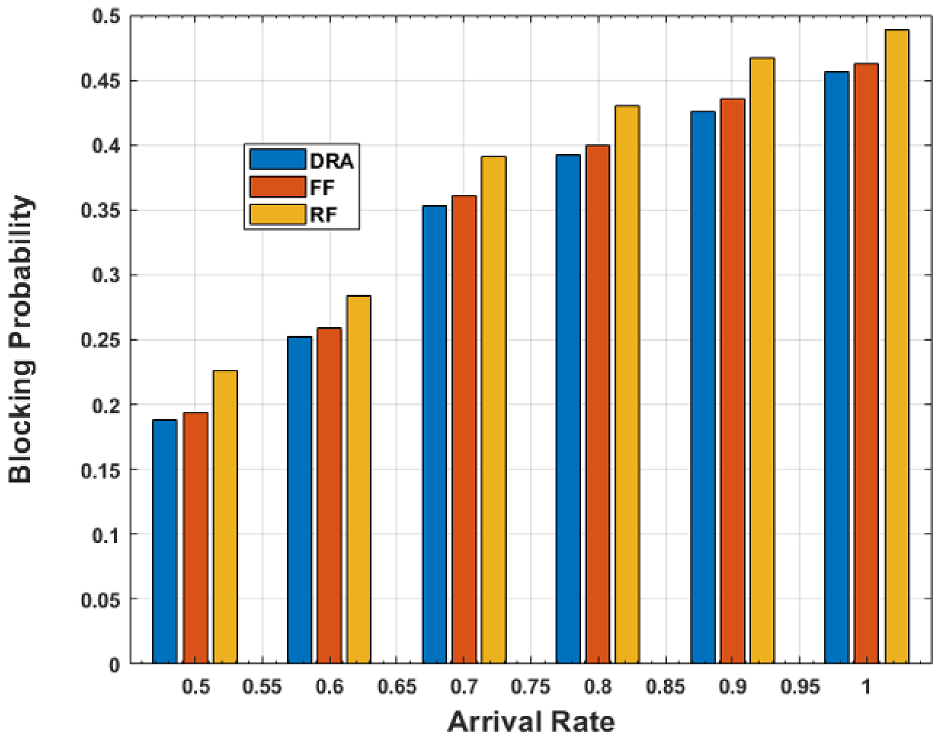


Figure 4.3: Testing results of BP versus arrival rate

From Figure 4.4 it can be observe that as traffic load increases in the network resource utilization also increases due to the accomodation of large number of requests. The proposed DRA approach utilize efficient resources of the network in order to serve the network resources as compared to the other two existing approaches in the literature as shown in Figure 4.3. The improvement in RU with DRA is 1.19% and 4.1% compared to FF and RF, respectively.

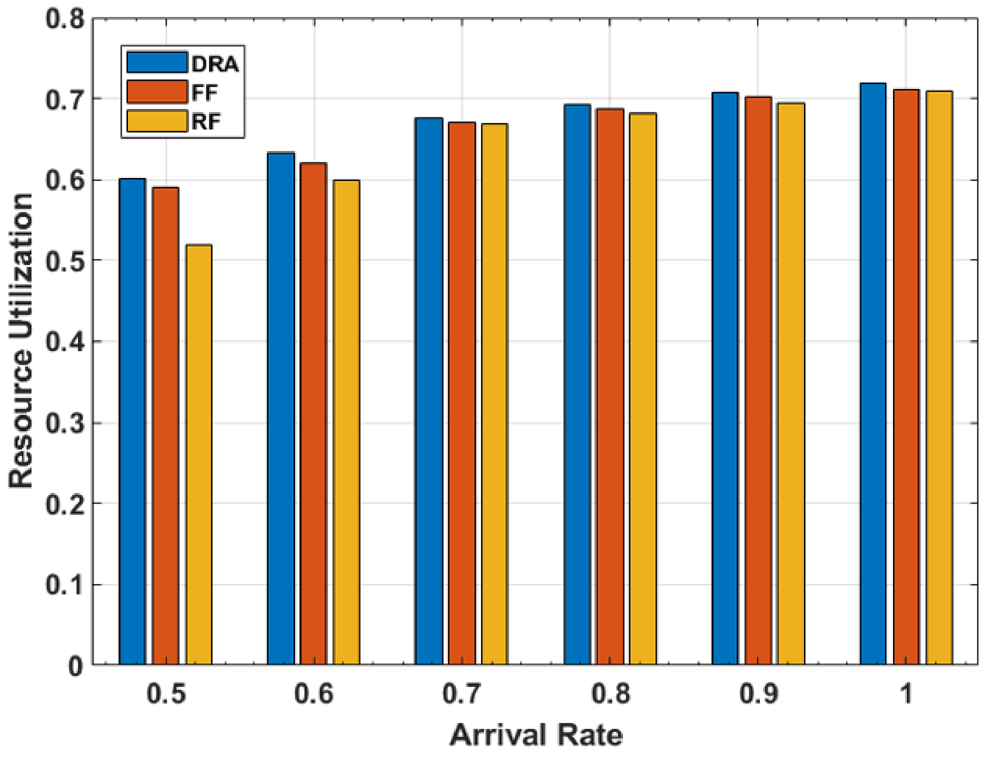


Figure 4.4: Testing results of RU versus arrival rate

Chapter 5

Conclusion and Scope for Future Work

Artificial Narrow Intelligence (ANI) is being used in every field of science to address the challenges of various applications such as healthcare, finance, robotics, autonomous driving, drones warfare, resource and logistic management.Machine Learning (ML), Deep Learning (DL) and big data are the backbone of today's AI techniques.

Nowadays, we are shifting towards Artificial General Intelligence (AGI) using the immense potential of deep reinforcement learning (DRL) to solve sequential decision making problems, which uses experience based learning.

In this work, we have explored DRL to optimize the routing and resources assignment in QKD network while establishing the requests. Simulation results indicate that the proposed DRA approach perform better in terms of blocking probability as compared to baseline approach after sufficient training iterations using model-free DQN and A2C algorithms.

In future, model-based DRL algorithms can be explored in the same environment to address various challenges.In addition to this, we can also apply DRL techniques in autonomous driving for 3D object detection and path planning.

References

[1] T. Miller, “Explanation in artificial intelligence: Insights from the social sciences,” *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[2] R. Dazeley, P. Vamplew, C. Foale, C. Young, S. Aryal, and F. Cruz, “Levels of explainable artificial intelligence for human-aligned conversational explanations,” *Artificial Intelligence*, vol. 299, p. 103525, 2021.

[3] S. Angra and S. Ahuja, “Machine learning and its applications: A review,” in *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC).* IEEE, 2017, pp. 57–60.

[4] T. Mitchell and M. L. McGraw-Hill, “Edition,” 1997.

[5] J. DSouza *et al.*, “Using exploratory data analysis for generating inferences on the correlation of covid-19 cases,” in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT).* IEEE, 2020, pp. 1–6.

[6] B. Mukherjee, “WDM optical communication networks: progress and challenges,” *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1810–1824, Oct. 2000.

[7] Y. Zhao, Y. Cao, W. Wang, H. Wang, X. Yu, J. Zhang, M. Tornatore, Y. Wu, and B. Mukherjee, “Resource allocation in optical networks secured by quantum key distribution,” *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 130–137, 2018.

[8] Y. Cao, Y. Zhao, X. Yu, and Y. Wu, “Resource assignment strategy in optical networks integrated with quantum key distribution,” *IEEE/OSA J. Opt. Commun. Netw.*, vol. 9, no. 11, pp. 995–1004, 2017.

[9] Y. Cao, Y. Zhao, X. Yu, H. Wang, C. Liu, B. Li, and J. Zhang, “Resource allocation in software-defined optical networks secured by quantum key distribution,” in *Proc. IEEE OECC/PGC*, Singapore, 2017, pp. 1–3.

[10] P. Jouguet, S. Kunz-Jacques, T. Debuisschert, S. Fossier, E. Diamanti, R. Alléaume, R. Tualle-Brouri, P. Grangier, A. Leverrier, P. Pache *et al.*, “Field

test of classical symmetric encryption with continuous variables quantum key distribution," *Opt. Express*, vol. 20, no. 13, pp. 14 030–14 041, 2012.

[11] N. Gisin, G. Ribordy, W. Tittel, and H. Zbinden, "Quantum cryptography," *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 145–195, Mar. 2002.

[12] P. Sharma, A. Agrawal, V. Bhatia, S. Prakash, and A. K. Mishra, "Quantum key distribution secured optical networks: A survey," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 2049–2083, Sep. 2021.

[13] Y. Cao, Y. Zhao, Y. Wu, X. Yu, and J. Zhang, "Time-scheduled quantum key distribution (QKD) over WDM networks," *IEEE/OSA J. Lightw. Technol.*, vol. 36, no. 16, pp. 3382–3395, Aug. 2018.

[14] H. Wang, Y. Zhao, X. Yu, Z. Ma, J. Wang, A. Nag, L. Yi, and J. Zhang, "Protection schemes for key service in optical networks secured by quantum key distribution (QKD)," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 11, no. 3, pp. 67–78, 2019.