A PROJECT REPORT

ON

"IMPLEMENTATION OF MACHINE LEARNING IN IoT"

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

BACHELOR ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

UNDER

DIBRUGARH UNIVERSITY

Submitted by:

AMIT GOYAL	**CS-19/16**
AKASHDEEP NANDI	**CS-05/16**
INDRAJIT SHARMA	**CS-09/16**
INDRAJIT SEN	**CS-29/16**
NITESH KR PATHAK	**CS-24/16**

Under the guidance of

MR. SYED IBTISAM TAUHIDI

Assistant Professor

DEPARTMENT OF COMPUTER AND SCIENCE ENGINEERING

JORHAT ENGINEERING COLLEGE

JORHAT-785007

ACKNOWLEDGEMENT

Prima facie, we are grateful to God for the good health and well-being that were necessary to complete this project.

We wish to express our sincere thanks to Dr. Rupam Baruah, Head of the Department, Computer Science and Engineering, for providing us with all the necessary means for completion of the project.

We are also grateful to Mr. Syed Ibtisam Tauhidi, Assistant Professor, in the Department of Computer Science and Engineering. We are extremely thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement extended to us.

We also place on record, our sense of gratitude to one and all, who directly or indirectly, have lent their hand in this project.

AMIT GOYAL	CS-19/16
AKASHDEEP NANDI	CS-05/16
INDRAJIT SHARMA	CS-09/16
INDRAJIT SEN	CS-29/16
NITESH KR PATHAK	CS-24/16

Place: Jorhat

Date: 06/08/2020

CERTIFICATE

JORHAT ENGINEERING COLLEGE
DIBRUGARH UNIVERSITY

GOVERNMENT OF ASSAM
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JORHAT ENGINEERING COLLEGE
JORHAT-7, ASSAM

This is to certify that the project entitled "**Implementation of Machine Learning in IoT**" submitted by

AMIT GOYAL CS-19/16
AKASHDEEP NANDI CS-05/16
INDRAJIT SHARMA CS-09/16
INDRAJIT SEN CS-29/16
NITESH KR PATHAK CS-24/16

in fulfilment of the requirements for the Final Year Project in Computer Science and Engineering under Dibrugarh University has been carried out under my guidance. The contents of this report have not been fulfilled in part or full to any other university for the award of any other degree or diploma.

Date: 06/08/2020

(Dr. Rupam Baruah)
Head of the Department
Department of Computer Science and Engineering Jorhat Engineering College
Jorhat-7, Assam

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the project entitled “RECOMMENDATION ENGINE” is submitted by:

AMIT GOYAL	CS-19/16
AKASHDEEP NANDI	CS-05/16
INDRAJIT SHARMA	CS-09/16
INDRAJIT SEN	CS-29/16
NITESH KR PATHAK	CS-24/16

8th semester, COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of the requirements for the award of degree in BACHELOR OF ENGINEERING under DIBRUGARH UNIVERSITY has been carried out under my guidance.

The contents of this report have not been submitted to any other university for the award of any other degree or diploma.

Date: 06/08/2020

Place: Jorhat

Mr. Syed Ibtisam Tauhidi

Assistant professor

Computer Science and Engineering, Jorhat Engineering College.

Jorhat-785007, Assam

Contents

Chapter 1

Introduction

1.1 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust their actions accordingly.

1.1.1 Some machine learning methods

1. **Supervised machine learning** algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

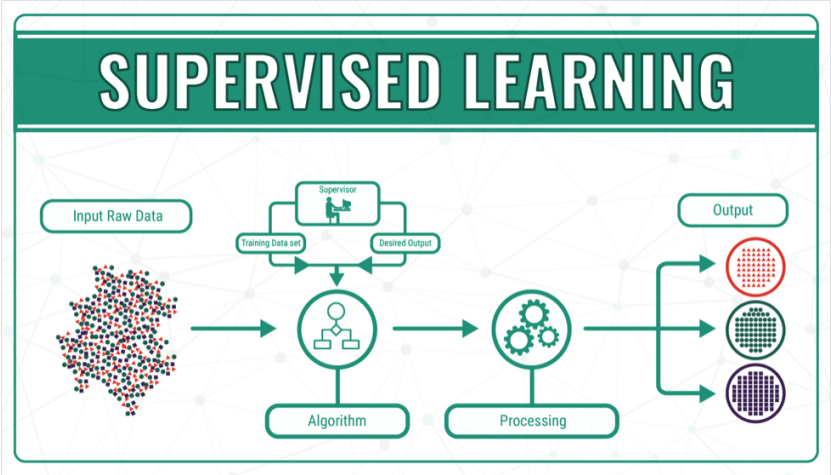


Figure 1.1: Supervised Learning

2. **Unsupervised machine learning** algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

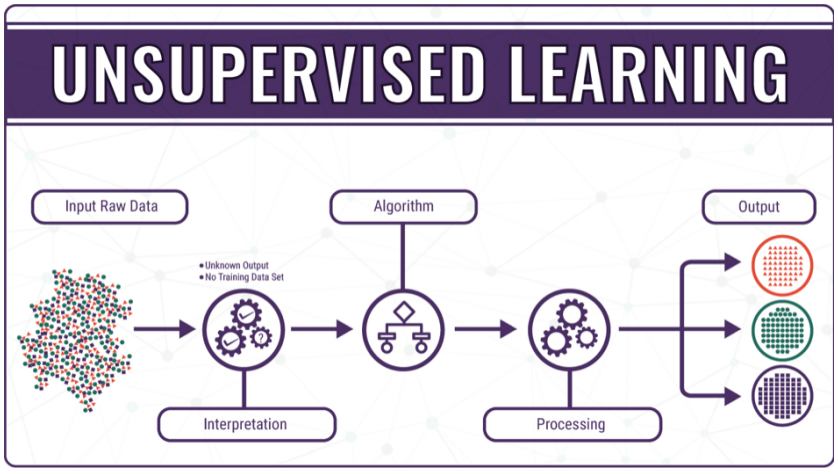


Figure 1.2: Unsupervised Learning

3. **Semi-supervised machine learning** algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of

labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

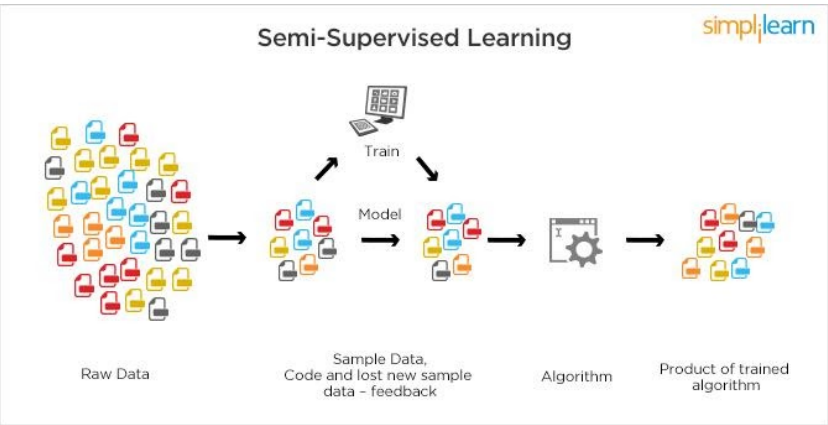


Figure 1.3: Semi-supervised machine learning

4. **Reinforcement machine learning** algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning.

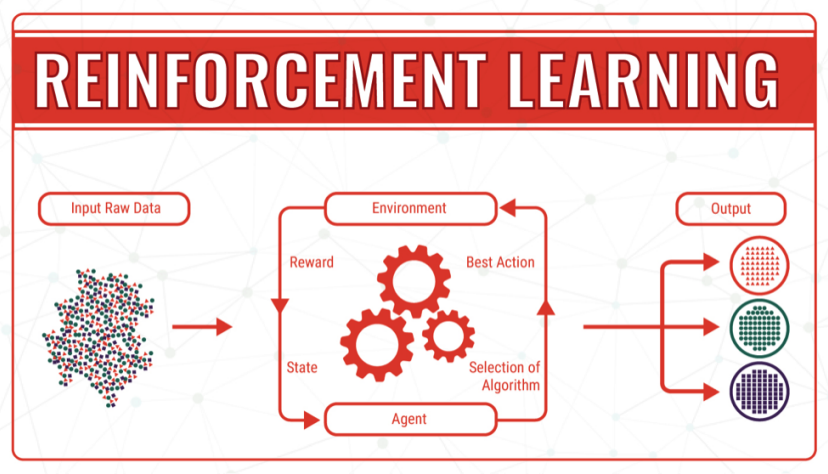


Figure 1.4: Reinforcement machine learnin

This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal. Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

1.1.2 Application of machine learning in day to day life

1. Virtual Personal Assistants

   Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice. All you need to do is activate them and ask, "What is my schedule for today?", "What are the flights from Germany to London", or similar questions. For answering, your personal assistant looks out for the information, recalls your related queries, or send a command to other resources (like phone apps) to collect info. You can even instruct assistants for certain tasks like "Set an alarm for 6 AM next morning", "Remind me to visit the visa office the day after tomorrow".

   Machine learning is an important part of these personal assistants as they collect and refine the information on the basis of your previous involvement with them. Later, this set of data is utilized to render results that are tailored to your preferences. Virtual Assistants are integrated to a variety of platforms. For example, devices like Amazon Echo and Google Home, and smartphones softwares like Samsung Bixby on Samsung S8

2. Predictions while Commuting

   Traffic Predictions: We all have been using GPS navigation services. While we do that, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic. While this helps in preventing the traffic and congestion analysis, the underlying problem is that there are less number of cars that are equipped with GPS. Machine learning in such

scenarios helps to estimate the regions where congestion can be found on the basis of daily experiences.

Online Transportation Networks: When booking a cab, the app estimates the price of the ride. When sharing these services, how do they minimize the detours? The answer is machine learning. Jeff Schneider, the engineering lead at Uber ATC reveals in an interview that they use ML to define price surge hours by predicting the rider demand. In the entire cycle of the services, ML is playing a major role.

3. Video Surveillance

   Imagine a single person monitoring multiple video cameras! Certainly, a difficult job to do and boring as well. This is why the idea of training computers to do this job makes sense.

   The video surveillance systems nowadays are powered by AI that makes it possible to detect crime before they happen. They track unusual behaviour of people like standing motionless for a long time, stumbling, or napping on benches etc. The system can thus give an alert to human attendants, which can ultimately help to avoid mishaps. And when such activities are reported and counted to be true, they help to improve the surveillance services. This happens with machine learning doing its job at the backend.

4. Social Media Services

   From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits. Here are a few examples that you must be noticing, using, and loving in your social media accounts, without realizing that these wonderful features are nothing but the applications of ML.

   People You May Know: Machine learning works on a simple concept: understanding with experiences. Facebook continuously notices the friends that you connect with, the profiles that you visit very often, your interests, workplace, or a group that you share with someone etc. On the basis of continuous learning, a list of Facebook users are suggested that you can become friends with.

   Face Recognition: You upload a picture of you with a friend and Facebook instantly recognizes that friend. Facebook checks the poses and projections in the picture, notice the unique features, and then match them with the people in your friend list. The entire process at the backend is complicated and takes care of the precision factor but seems to be a simple application of ML at the front end.

Similar Pins: Machine learning is the core element of Computer Vision, which is a technique to extract useful information from images and videos. Pinterest uses computer vision to identify the objects (or pins) in the images and recommend similar pins accordingly.

5. Email Spam and Malware Filtering

   There are a number of spam filtering approaches that email clients use. To ascertain that these spam filters are continuously updated, they are powered by machine learning. When rule-based spam filtering is done, it fails to track the latest tricks adopted by spammers. Multi Layer Perceptron, C 4.5 Decision Tree Induction are some of the spam filtering techniques that are powered by ML.

   Over 325, 000 malwares are detected everyday and each piece of code is 90–98% similar to its previous versions. The system security programs that are powered by machine learning understand the coding pattern. Therefore, they detect new malware with 2–10% variation easily and offer protection against them.

6. Online Customer Support

   A number of websites nowadays offer the option to chat with customer support representatives while they are navigating within the site. However, not every website has a live executive to answer your queries. In most of the cases, you talk to a chatbot. These bots tend to extract information from the website and present it to the customers. Meanwhile, the chatbots advance with time. They tend to understand the user queries better and serve them with better answers, which is possible due to its machine learning algorithms.

7. Search Engine Result Refining

   Google and other search engines use machine learning to improve the search results for you. Every time you execute a search, the algorithms at the backend keep a watch at how you respond to the results. If you open the top results and stay on the web page for long, the search engine assumes that the results it displayed were in accordance to the query. Similarly, if you reach the second or third page of the search results but do not open any of the results, the search engine estimates that the results served did not match the requirement. This way, the algorithms working at the backend improve the search results.

8. Product Recommendations

You shopped for a product online a few days back and then you keep receiving emails for shopping suggestions. If not this, then you might have noticed that the shopping website or the app recommends you some items that somehow match with your taste. Certainly, this refines the shopping experience but did you know that it's machine learning doing the magic for you? On the basis of your behaviour with the website/app, past purchases, items liked or added to cart, brand preferences etc., the product recommendations are made.

9. Online Fraud Detection

   Machine learning is proving its potential to make cyberspace a secure place and tracking monetary frauds online is one of its examples. For example: Paypal is using ML for protection against money laundering. The company uses a set of tools that helps them to compare millions of transactions taking place and distinguish between legitimate or illegitimate transactions taking place between the buyers and sellers.

1.2 Internet of Things

Internet of Things (IoT) is an ecosystem of connected physical objects that are accessible through the internet. The 'thing' in IoT could be a person with a heart monitor or an automobile with built-in-sensors, i.e. objects that have been assigned an IP address and have the ability to collect and transfer data over a network without manual assistance or intervention. The embedded technology in the objects helps them to interact with internal states or the external environment, which in turn affects the decisions taken.

1.2.1 Importance of IoT

When something is connected to the internet, that means that it can send information or receive information, or both. This ability to send and/or receive information, aided by appropriate software, makes things appear 'smart', which is a desirable feature.

Let's use smartphones as an example. Right now we can listen to just about any song in the world, despite the songs not being stored in our phones. It's because the requested song is stored somewhere else, which our phone can request (for example, by sending a HTTP request asking for that song) and then receive (streaming that song to your phone).

To be smart, a thing doesn't need to have a very high storage or processing capability. It just requires a connection with a computer which can provide

it such desired capability. The storage or processing of data can be off-loaded to such high-capability remote computers.

In the Internet of Things, all the things that are being connected to the internet can be put into three categories:

- Things that collect information and then transmit it,
- Things that receive information and then act on it,
- Things that do both.

And all three of these have enormous benefits that feed on each other

1. Collecting and Sending Information

   An example of collecting information is the sensing of the external environment through sensors like temperature sensors, motion sensors, moisture sensors, air quality sensors, light sensors, etc. These sensors, along with a connection, allow us to automatically collect information from the environment which, in turn, allows us to make more intelligent decisions.

   On the farm, automatically getting information about the soil moisture can tell farmers exactly when their crops need to be watered. Instead of watering too much (which can be an expensive over-use of irrigation systems and environmentally wasteful) or watering too little (which can be an expensive loss of crops), the farmer can ensure that crops get exactly the right amount of water. More money for farmers and more food for the world!

   Just as our sight, hearing, smell, touch, and taste allow us, humans, to make sense of the world, sensors allow machines to make sense of the world.

2. Receiving and Acting on Information

   We're all very familiar with machines receiving information and then acting on it. Your printer receives a document and it prints it. Your car receives a signal from your car keys and the doors open. The examples are endless.

   Whether it's as simple as sending the command "turn on" or as complex as sending a 3D model's STL file to a 3D printer, we know that we can tell machines what to do from far away.

3. Doing Both

   The real power of the Internet of Things arises when things can do both of the above. Things that collect information and send it, but also receive information and act on it.

   Let's take the farming example. The sensors can collect information about the soil moisture to tell the farmer how much to water the crops, but you don't actually need the farmer. Instead, the irrigation system can automatically turn on as needed, based on how much moisture is in the soil.

1.2.2 Technologies that made IoT possible

While the idea of IoT has been in existence for a long time, a collection of recent advances in a number of different technologies has made it practical.

- **Access to low-cost, low-power sensor technology**: Affordable and reliable sensors are making IoT technology possible for more manufacturers.
- Connectivity : A host of network protocols for the internet has made it easy to connect sensors to the cloud and to other "things" for efficient data transfer.
- **Cloud computing platforms**: The increase in the availability of cloud platforms enables both businesses and consumers to access the infrastructure they need to scale up without actually having to manage it all.
- **Machine learning and analytics**: With advances in machine learning and analytics, along with access to varied and vast amounts of data stored in the cloud, businesses can gather insights faster and more easily. The emergence of these allied technologies continues to push the boundaries of IoT and the data produced by IoT also feeds these technologies.
- **Conversational artificial intelligence (AI)**: Advances in neural networks have brought natural-language processing (NLP) to IoT devices (such as digital personal assistants Alexa, Cortana, and Siri) and made them appealing, affordable, and viable for home use.

1.3 Neural Network

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of intelligent systems.

Neural networks, in the world of finance, for example, assist in the development of such processes as time-series forecasting, algorithmic trading, securities classification, credit risk modeling and constructing proprietary indicators and price derivatives.

A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

A neural network contains layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that may be nonlinear.

In a multi-layered perceptron (MLP), perceptrons are arranged in interconnected layers. The input layer collects input patterns. The output layer has classifications or output signals to which input patterns may map. For instance, the patterns may comprise a list of quantities for technical indicators about a security; potential outputs could be "buy," "hold" or "sell."

Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

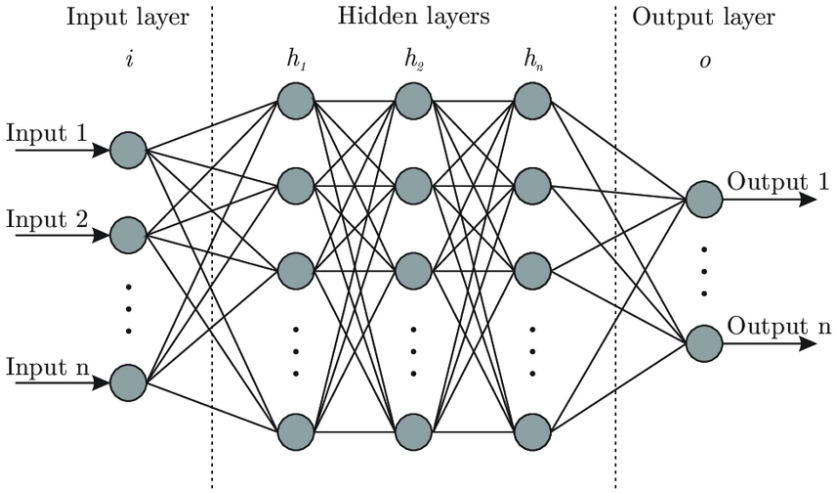


Figure 1.5: Multi-layer neural network

1.3.1 Areas of Application

Followings are some of the areas where ANN is being used. It suggests that ANN has an interdisciplinary approach in its development and applications.

- **Speech Recognition**

  Speech occupies a prominent role in human-human interaction. Therefore, it is natural for people to expect speech interfaces with computers. In the present era, for communication with machines, humans still need sophisticated languages which are difficult to learn and use. To ease this communication barrier, a simple solution could be, communication in a spoken language that is possible for the machine to understand.

  Great progress has been made in this field, however, still such kinds of systems are facing the problem of limited vocabulary or grammar along with the issue of retraining of the system for different speakers in different conditions. ANN is playing a major role in this area.

- **Character Recognition**

  It is an interesting problem which falls under the general area of Pattern Recognition. Many neural networks have been developed for automatic recognition of handwritten characters, either letters or digits.

  Though back-propagation neural networks have several hidden layers, the pattern of connections from one layer to the next is localized. Similarly, neocognitron also has several hidden layers and its training is done layer by layer for such kind of applications.

- **Signature Verification Application**

  Signatures are one of the most useful ways to authorize and authenticate a person in legal transactions. Signature verification technique is a non-vision based technique. For this application, the first approach is to extract the feature or rather the geometrical feature set representing the signature. With these feature sets, we have to train the neural networks using an efficient neural network algorithm. This trained neural network will classify the signature as being genuine or forged under the verification stage.

- **Human Face Recognition**

  It is one of the biometric methods to identify the given face. It is a typical task because of the characterization of "non-face" images. However, if a neural network is well trained, then it can be divided into two classes namely images having faces and images that do not have faces.

  First, all the input images must be preprocessed. Then, the dimensionality of that image must be reduced. And, at last it must be classified using neural network training algorithms.

1.3.2 Types of Neural networks

There are many types of neural nets available or that might be in the development stage. They can be classified depending on their: Structure, Data flow, Neurons used and their density, Layers and their depth activation filters etc. The following are some types of neural network.

1. **Feed Forward Neural Networks**: Simplest form of neural nets where input data travels in one direction only, passing through artificial neural nodes and exiting through output nodes. Where hidden layers may or may not be present, input and output layers are present there. Based on this, they can be further classified as single layered or multi-layered feed forward neural nets. Number of layers depends on the complexity of the function.

2. **Multi-Layer Perceptron**: An entry point towards complex neural nets where input data travels through various layers of artificial neurons. Every single node is connected to all neurons in the next layer which makes it a fully connected neural network. Input and output layers are present having multiple hidden Layers i.e. at least three

or more layers in total. It has a bi-directional propagation i.e. forward propagation and backward propagation. Inputs are multiplied with weights and fed to the activation function and in back propagation they are modified to reduce the loss. In simple words, weights are machine learnt values from Neural Networks.

3. **Convolution Neural Network**: Convolution neural nets contains a three-dimensional arrangement of neurons, instead of the standard two-dimensional array. The first layer is called a convolutional layer. Each neuron in the convolutional layer only processes the information from a small part of the visual field. Input features are taken in batch wise like a filter. Network understands the images in parts and can compute these operations multiple times to complete the full image processing. Processing involves conversion of the image from RGB or HSI scale to gray-scale.

4. **Radial Basis Function Neural Networks**: Radial Basis Function Network consists of an input vector followed by a layer of RBF neurons and an output layer with one node per category. Classification is performed by measuring the input's similarity to data points from the training set where each neuron stores a prototype. This will be one of the examples from the training set. When a new input vector [the n-dimensional vector that you are trying to classify] needs to be classified, each neuron calculates the Euclidean distance between the input and its prototype. For example, if we have two classes i.e. class A and Class B, then the new input to be classified is more close to class A prototypes than the class B prototypes. Hence, it could be tagged or classified as class A.

1.4 Deep Neural Networks

A Deep Neural Network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship.

1.4.1 Deep Learning

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial

intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

1.4.2 Machine Learning vs Deep Learning

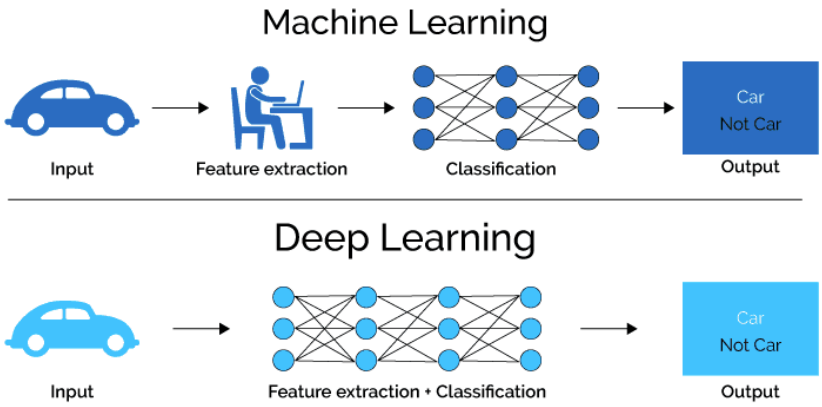


Figure 1.6: Machine Learning vs Deep Learning

One of the most common AI techniques used for processing big data is machine learning, a self-adaptive algorithm that gets increasingly better analysis and patterns with experience or with newly added data.

If a digital payments company wanted to detect the occurrence or potential for fraud in its system, it could employ machine learning tools for this purpose. The computational algorithm built into a computer model will process all transactions happening on the digital platform, find patterns in the data set and point out any anomaly detected by the pattern.

Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected together like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

A traditional approach to detecting fraud or money laundering might rely on the amount of transactions that ensues, while a deep learning nonlinear technique would include time, geographic location, IP address, type of retailer and any other feature that is likely to point to fraudulent activity. The first layer of the neural network processes a raw data input like the amount of the transaction and passes it on to the next layer as output. The second layer

processes the previous layer's information by including additional information like the user's IP address and passes on its result.

The next layer takes the second layer's information and includes raw data like geographic location and makes the machine's pattern even better. This continues across all levels of the neuron network.

1.5 Backpropagation

Backpropagation, short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights. It is a generalization of the delta rule for perceptrons to multilayer feedforward neural networks.

The "backwards" part of the name stems from the fact that calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow of the error information allows for efficient computation of the gradient at each layer versus the naive approach of calculating the gradient of each layer separately.

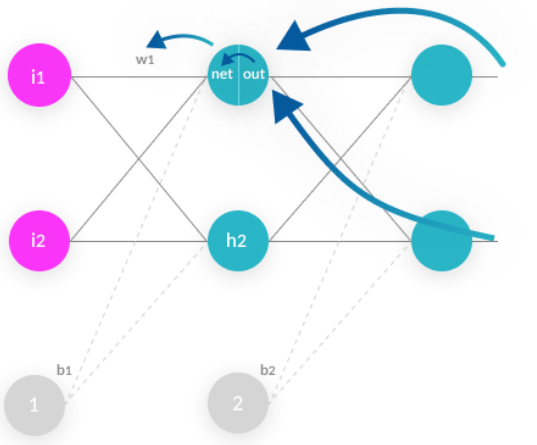


Figure 1.7: Representation of backpropagation in a network.

1.6 Gradient Descent

Gradient descent is a first order iterative optimization algorithm for finding the minimum of a function. The goal of the gradient descent is to minimise a given function which, in our case, is the loss function of the neural network.

To achieve this goal, it performs two steps iteratively -

1. Compute the slope (gradient) that is the first-order derivative of the function at the point.

2. Move-in the opposite direction of the slope increases from the current point by the computed amount.

So, the idea is to pass the training set through the hidden layers of the neural network and then update the parameters of the layers by computing the gradients using the training samples from the training dataset.

For example, suppose that someone is at the top of a mountain, and has to reach a valley which is at the lowest point of the mountain, while being blindfolded and having zero visibility to see where she headed. So, what approach should one take to reach the valley?

The best way is to check the ground near oneself and observe where the land tends to descend. This will give an idea in what direction one should take the next step. If one follows the descending path, it is very likely she would reach the valley. This is represented graphically in Fig. 1.8.

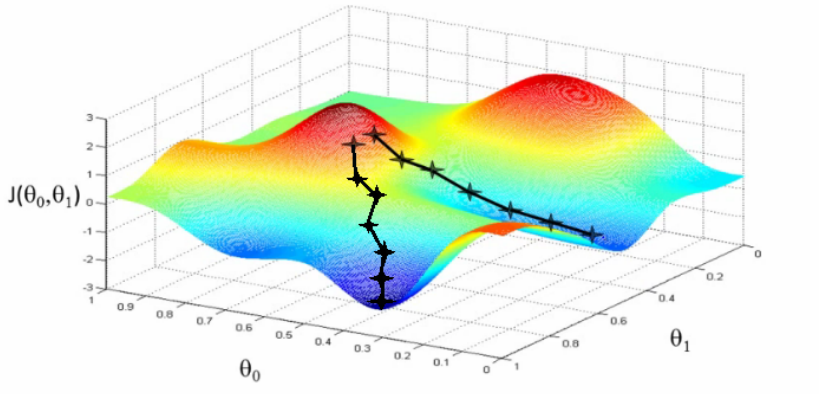


Figure 1.8: Gradient descent on a 2D error surface

Let us now map this scenario with fig 1.8. Suppose we want to find out the best parameters 0 and 1 for our learning algorithm. Similar to the analogy above, we see we find similar mountains and valleys when we plot our "cost

space". Cost Space is nothing but how our algorithm would perform when we choose a particular value for a parameter.

So on the z-axis, we have the cost J(0,1) against our parameters 0 and 1 on x-axis and y-axis respectively. Here, hills are represented by red regions, which have high cost, and valleys are represented by blue regions, which have low cost.

1.7 Types of Gradient Descent

1.7.1 Batch Gradient Descent

In Batch Gradient Descent, all the training data is taken into consideration to take a single step. We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters. So that's just one step of gradient descent in one epoch.

Batch Gradient Descent is great for convex or relatively smooth error manifolds. In this case, we move directly towards an optimum solution.

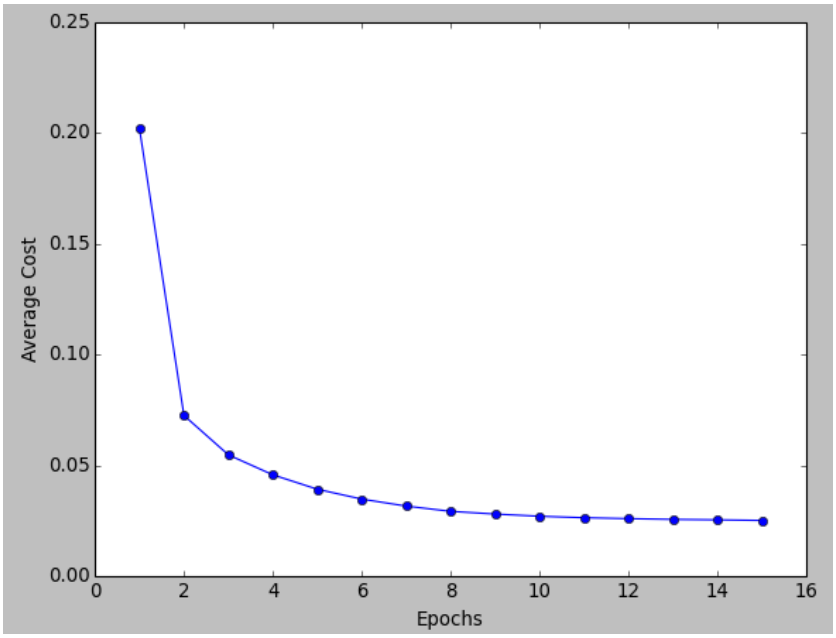


Figure 1.9: Batch Gradient Descent

The above figure 1.7.1.1 in which the graph of cost vs epochs is also quite smooth because we are averaging over all the gradients of training data for a single step. The cost keeps on decreasing over the epochs.

1.7.2 Stochastic Gradient Descent

In Batch Gradient Descent we were considering all the examples for every step of Gradient Descent. But this is very time-expensive per epoch. Suppose our dataset has 5 million examples, then just to take one step the model will have to calculate the gradients of all the 5 million examples. This increases the time taken to calculate one step of the gradient descent. To tackle this problem we have Stochastic Gradient Descent. In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step. We do the following steps in one epoch for SGD:

1. Take an example.
2. Feed it to Neural Network.
3. Calculate its gradient.
4. Use the gradient we calculated in step 3 to update the weights.
5. Repeat steps 1–4 for all the examples in training dataset.

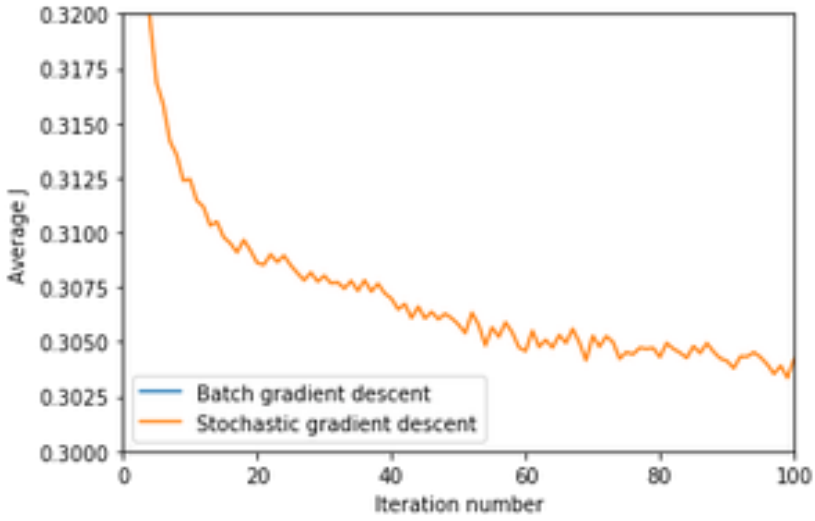


Figure 1.10: Stochastic Gradient Descent

Since we are considering just one example at a time the cost will fluctuate over the training examples and it will not necessarily decrease. But in the long run, you will see the cost decreasing with fluctuations.

Also because the cost is so fluctuating, it will never reach the minima but it will keep oscillating around it. SGD can be used for larger datasets. It converges faster when the dataset is large as it causes updates to the parameters more frequently.

1.7.3 Mini Batch Gradient Descent

We have seen the Batch Gradient Descent. We have also seen the Stochastic Gradient Descent. Batch Gradient Descent can be used for smoother curves. SGD can be used when the dataset is large. Batch Gradient Descent converges directly to minima. SGD converges faster for larger datasets. But, since in SGD we use only one example at a time, we have higher oscillations when approaching the minima. This can slow down the computations by increasing the numbers of epochs required. To tackle this problem, a mixture of Batch Gradient Descent and SGD is used.

Neither do we use all the dataset all at once nor do we use the single example at a time. We use a batch of a fixed number of training examples which is less than the actual dataset and call it a mini-batch. Doing this helps us achieve the advantages of both the former variants we saw. So, after creating the mini-batches of fixed size, we do the following steps in one epoch:

1. Pick a mini-batch.
2. Feed it to Neural Network.
3. Calculate the mean gradient of the mini-batch.
4. the mean gradient we calculated in step 3 to update the weights.
5. Repeat steps 1–4 for the mini-batches we created.

Just like SGD, the average cost over the epochs in mini-batch gradient descent fluctuates because we are averaging a small number of examples at a time.

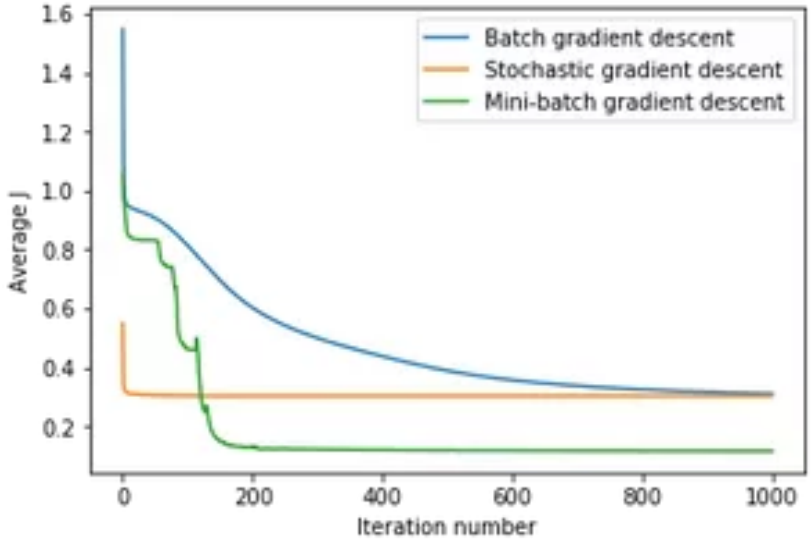


Figure 1.11: Mini Batch Gradient Descent

1.8 Online Training

Online training is sometimes the only viable option for dealing with large amounts of data. Specifically, when you are seeing so much data that you only get a chance to look at each data point once, online training can be a good option. Also, online training is necessary when training on real-time data.

Batch learning and online learning both have their place. Generally speaking batch learning will train your neural network to a lower residual error level, because the online training can sometimes have one training undo the effect of another.

Considering the limitations (storage and processing) of IoT devices in our project and as we are dealing with real time data, we simulate the data sending of the IoT devices with already created static datasets and we send the data one record at a time from each dataset to cloudMQTT and from there we fetch the records to the training nodes.

There are few advantages of online training -

1. Online learning schemes learn "faster." In some cases, determining how many samples you need can be difficult, and sample gathering may be costly. Online learning lets you see the progress of your training as the number of samples increases, potentially saving on sample gathering costs once it reaches an acceptable error.

2. Online learning schemes can be done during operation, which can be valuable.

3. Online learning schemes are more effective for dealing with data sets that are not stationary. It is easier for the batch methods to come up with highly specialized solutions which work well for the test samples, but do not capture a general solution.

Hence, online training processes one sample at a time and can thus be significantly more efficient both in time and space and more practical than batch learning.

1.9 MQTT

MQTT is a simple messaging protocol, designed for constrained devices with low-bandwidth. So, it's the perfect solution for Internet of Things applications. MQTT allows you to send commands to control outputs, read and publish data from sensor nodes and much more. Therefore, it makes it really easy to establish a communication between multiple devices.

There are few basic concepts regarding MQTT:

- **MQTT – Publish/Subscribe**

In a publish and subscribe system, a device can publish a message on a particular topic, or it can be subscribed to a particular topic to receive messages.

- **MQTT – Messages**

Messages are the information that you want to exchange between your devices. Whether it's a command or data.

- **MQTT – Topics**

Topics are the way you register interest for incoming messages or how you specify where you want to publish the message.

- **MQTT – Broker**

  The broker is primarily responsible for receiving all messages, filtering the messages, deciding who is interested in them and then publishing the message to all subscribed clients.

  There are several brokers that one can use. Mosquitto is an open-source implementation of a broker which can be installed locally in our own PCs or microcontrollers like the Raspberry Pi. Alternatively, one can use CloudMQTT, which is a cloud-based broker.

1.9.1 CloudMQTT

CloudMQTT is a managed Mosquitto server in the cloud. Mosquitto implements the MQ Telemetry Transport protocol, MQTT, which provides lightweight methods of carrying out messaging using a publish/subscribe message queueing model.

CloudMQTT let you focus on the application instead of spending time on scaling the broker or patching the platform.

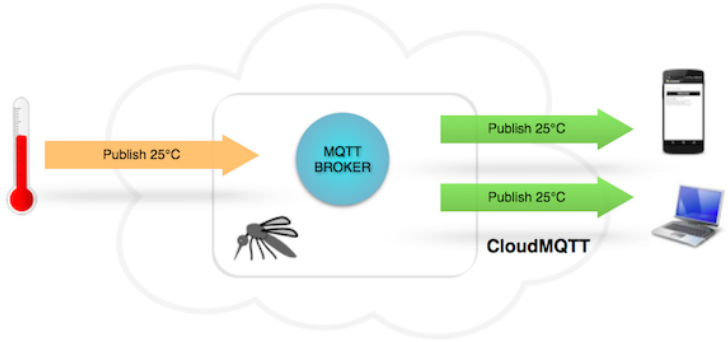


Figure 1.12: CloudMQTT working model

Chapter 2

Literature Survey

2.1 Abstract

Rapid developments in hardware, software, and communication technologies have allowed the emergence of Internet-connected sensory devices that provide observation and data measurement from the physical world. By 2020, it is estimated that the total number of Internet-connected devices being used will be between 25-50 billion. As the number grows and technologies become more mature, the volume of data published will increase. Internet-connected devices technology, referred to as Internet of Things (IoT), continues to extend the current Internet by providing connectivity and interaction between the physical and cyber worlds. In addition to increased volume, the IoT generates Big Data characterized by velocity in terms of time and location dependency, with a variety of multiple modalities and varying data quality. Intelligent processing and analysis of this Big Data is the key to developing smart IoT applications. This article assesses the different machine learning methods that deal with the challenges in IoT data by considering smart cities as the main use case. The key contribution of this study is the presentation of a taxonomy of machine learning algorithms explaining how different techniques are applied to the data in order to extract higher level information.

2.2 Introduction

Emerging technologies in recent years and major enhancements to Internet protocols and computing systems, have made the communication between different devices easier than ever before. According to various forecasts, around 25-50 billion devices are expected to be connected to the Internet by 2020. This has given rise to the newly developed concept of the Internet of

Things (IoT).

According to a research paper "Machine Learning for IoT data analysis", IoT is a combination of embedded technologies regarding wired and wireless communications, sensor and actuator devices, and the physical objects connected to the Internet .

IoT needs data to either represent better services to users or enhance IoT framework performance to accomplish this intelligently. In this manner, systems should be able to access raw data from different resources over the network and analyze this information to extract knowledge. Since IoT will be among the greatest sources of new data, data science will make a great contribution to make IoT applications more intelligent.

The purpose of the Internet of Things, (IoT) is to develop a smarter environment, and a simplified life-style by saving time, energy, and money. Through this technology, the expenses in different industries can be reduced. The enormous investments and many studies running on IoT has made IoT a growing trend in recent years. IoT is a set of connected devices that can transfer data among one another in order to optimize their performance; these actions occur automatically without human interference. IoT includes four main components: (1) sensors, (2) processing networks, (3) analyzing data, and (4) monitoring the system. The most recent advances made in IoT began when radio frequency identification (RFID) tags were put into use more frequently, lower cost sensors became more available, web technology developed, and communication protocols changed . The IoT is integrated with different technologies and connectivity is a necessary and sufficient condition for it. So communication protocols are constituents the technology that should be enhanced.

In IoT, communication protocols can be divided into three major components:

1. Device to Device (D2D): This type of communication enables communication between nearby mobile phones. This is the next generation of cellular networks.

2. Device to Server (D2S): In this type of communication devices, all the data is sent to the servers, which can be close or far from the devices. This type of communication is mostly applied in cloud processing.

3. Server to Server (S2S): In this type of communication, servers transmit data between each other. This type of communication is mostly applied in cellular networks.

Processing and preparing data for these communications is a critical challenge. To respond to this challenge, different kinds of data processing, such

as analytics at the edge, stream analysis, and IoT analysis at the database, must be applied. The decision to apply any one of the mentioned processes depends on 165 the particular application and its needs. Fog and cloud processing are two analytical methods adopted in processing and preparing data before transferring to the other things. The whole task of IoT is summarized as follows: first, sensors and IoT devices collect information from the environment. Next, knowledge should be extracted from the raw data. Then, the data will be ready for transfer to other objects, devices, or servers through the Internet [1].

One important challenge of implementing machine learning with IoT is the way data will be transferred to the nodes for training. In order to address this problem we have used the protocol MQTT (Message Queue Telemetry Transport). It is a publish/subscribe messaging transport protocol based on client-server architecture designed for M2M and IoT applications for constrained networks [2].

The main objective of our project is to answer the following questions -

1. How could machine learning algorithms be applied to IoT smart data or real world data?

2. What are IoT data characteristics in real-world?

To understand which algorithm is more appropriate for processing and decision-making on generated smart data from the things in IoT, realizing these three concepts is essential. First, the IoT application , second, the IoT data characteristics and the third, the data-driven vision of machine learning algorithms.

2.3 Machine Learing in IoT

Our increasingly connected world, combined with low-cost sensors and distributed intelligence, will have a transformative impact on industry, producing more data than humans will be able to process. Will businesses be able to adapt and evolve quickly enough to maintain their place in the competitive landscape? How will humans make sense of and benefit from these new sources of information and intelligence embedded in our environment?

Organizations will need to get their internal data houses in order so they can leverage new sources and streams of data. Smart connected devices will also remove humans from the loop in some cases, so devices will be making their own decisions and self-adjusting or course correcting and repairing themselves as needed. In other cases, collections of devices will act as systems that can be optimized in new ways, and systems of systems will share

data and behave as an ecosystem of data and devices. Machine learning — a term that describes numerous approaches to deriving meaning from data — will have to be part of the equation, but so will traditional business and data analysis techniques as organizations prepare for the Internet of Things (IoT) [4].

2.4 Deep Learning in IoT

Deep learning is a promising approach for extracting accurate information from raw sensor data from IoT devices deployed in complex environments. Deep learning can enable Internet of Things (IoT) devices to interpret unstructured multimedia data and intelligently react to both user and environmental events but has demanding performance and power requirements. Machine learning (ML) is applicable to many of the use cases that surround IoT products.

In recent years, increasing numbers of Internet of Things (IoT) products have appeared on the market that capture environmental data and use traditional machine-learning technologies to understand it. An example is Google's Nest Learning Thermostat, which records temperature data in a structured way and then applies algorithms to understand the patterns of its users' temperature preferences and schedules. However, it doesn't understand unstructured multimedia data, such as audio signals and visual images.

Emerging IoT devices apply more sophisticated deep-learning technologies that use neural networks to capture and understand their environments. Amazon Echo, for example, can understand human voice commands. It converts audio signals into a list of words, and then uses these words to search for relevant information. More recently, Microsoft's Windows IoT team released a face-recognition security system that utilizes deep-learning technology to perform tasks such as unlocking a door when it recognizes its users' faces.

Deep-learning applications on IoT devices often have demanding realtime requirements. For example, security camera–based object-recognition tasks usually require a detection latency of less than 500 ms to capture and respond to target events—for example, strangers appearing inside a house— in a timely manner. Commercial smart IoT devices often offload intelligence to the cloud. Nonetheless, consistent good-quality network connections, which are only available at limited locations and at high cost, become a challenging constraint for these devices to abide by real-time requirements. A better choice is to enable deep learning on the device, which isn't affected by the connection quality [4].

2.5 MQTT: A publish/subscribe protocol for wireless sensor network

In the past few years, Wireless Sensor Networks (WSNs) have been gaining increasing attention, both from a commercial and technical point of view, because of their potential of enabling novel and attractive solutions in areas such as industrial automation, asset management, environmental monitoring, transportation business, etc. Many of these applications require the transfer of data collected by the sensors to applications residing on a traditional network infrastructure (e.g Internet, LAN, enterprise network, etc.). Thus the WSNs need to be integrated with these traditional networks. Within the WSNs, a large number of battery-operated Sensor/Actuator (SA) devices, usually equipped with a limited amount of storage and processing capabilities, collect information about their environment and send them to the gateways for further transfer to the applications. Even for networks without actuators, information also flows in the opposite direction, e.g., for sensor management and configuration as well as for software updates.

The principle of the publish/subscribe (pub/sub) communication model is that components which are interested in consuming certain information register their interest. This process of registering an interest is called subscription, the interested party is therefore called a subscriber. Components which want to produce certain information do so by publishing their information. They are thus called publishers. The entity which ensures that the data gets from the publishers to the subscribers is the broker. The broker coordinates subscriptions, and subscribers usually have to contact the broker explicitly to subscribe.

There are three principal types of pub/sub systems: topic based, type-based and content-based. With topic-based systems, the list of topics is usually known in advance, e.g., during the design phase of an application. Subscriptions and publications can only be made on a specified set of topics. In type-based systems, a subscriber states the type of data it is interested in (e.g., temperature data). Type-based systems are not very common. Content-based systems are the most versatile ones. The subscriber describes the content of the messages it wants to receive. Such a subscription could be for any messages containing both temperature and light readings where the temperature is below a certain threshold and the light is on.

MQTT is a topic-based pub/sub protocol that uses character strings to provide support of hierarchical topics. This also facilitates the subscription to multiple topics. For example, a temperature sensor located on floor “F2”, room “R248” could publish its data using the hierarchi-

cal topic "wsn/sensor/F2/R248/temperature". The forward slash character "/" is used to separate each part of the topic. Wildcard characters can then be used to replace any part of the topic, e.g., the string "wsn/sensor/F2/+/temperature" could be employed to subscribe to data generated by all temperature sensors on floor F2. In this example the character "+" was used as a wildcard for any pattern at the 4th level of the topic. MQTT supports basic end-to-end Quality of Service (QoS). Depending on how reliably messages should be delivered to their receivers, MQTT distinguishes between three QoS levels. QoS level 0 is the simplest one: it offers a besteffort delivery service, in which messages are delivered either once or not at all to their destination. No retransmission or acknowledgment is defined. QoS level 1 provides a more reliable transport: messages with QoS level 1 are retransmitted until they are acknowledged by the receivers. Consequently, QoS level 1 messages are certain to arrive, but they may arrive multiple times at the destination because of retransmissions. The highest QoS level, QoS level 2, ensures not only the reception of the messages, but also that they are delivered only once to the receiving entities. It is up to the application to select the appropriate QoS level for its publications and subscriptions. For example, a temperature monitoring application could decide to use QoS level 0 for the publication of normal and regular measurement reports, but QoS level 1 for transferring alarm messages when the temperature exceeds a certain threshold. MQTT is a connection-oriented protocol in the sense that it requires a client to setup a connection with the broker before it can exchange publications and subscriptions with the broker. To this end, a "CONNECT" message is defined. It contains, among other connection parameters, a Client Id, which enables the broker to identify the connected client. This Client Id is used by the broker, for example to make sure that QoS level 1 and 2 publications are delivered correctly when the client reconnects after a network failure. The broker supervises the liveliness of the client/connection by a "keep-alive" timer, which defines the maximum time interval that may elapse between two messages received from that client. If during this time interval the client has no data-related messages to be transmitted, it will send a PING message to the broker, which is acknowledged by the broker. Thus the keep-alive timer enables the broker to detect the failure of either the client or the network link. A related and interesting MQTT feature is its support of the so-called "Will" concept. At connection time, a client could ask the broker to store a "Will" message together with a "Will" topic. The broker will only send this "Will" publication to the subscribers when it abnormally loses the connection with the client. Applications could use this feature to detect failures of devices and links [5].

Chapter 3

Methodology

3.1 Overview

The idea behind this project was to create an architecture for IoT devices which can learn adaptive ways to work on real time data.

For example, we have a swarm of IoT devices installed in an industrial setting with a number of sensors (luminosity, humidity, temperature, wind, radiation level, etc.) that is used to monitor the environment of the industry.

3.2 Approach

Our approach for this project can be pointed out in these steps -

1. Assembling an IoT device with sensors to collect data.
2. Send the real time data using a protocol over internet i.e. MQTT protocol.
3. Fetch the real time data in our training nodes.
4. Wait for the minimum number of records set in our Mini-Batch Gradient descent algorithm to be received in our training nodes.
5. Train on the data received and validate it to find accuracy.
6. The model trained providing best accuracy (best fit) sends the weight matrix to the IoT devices which can now work on their own to make decisions.

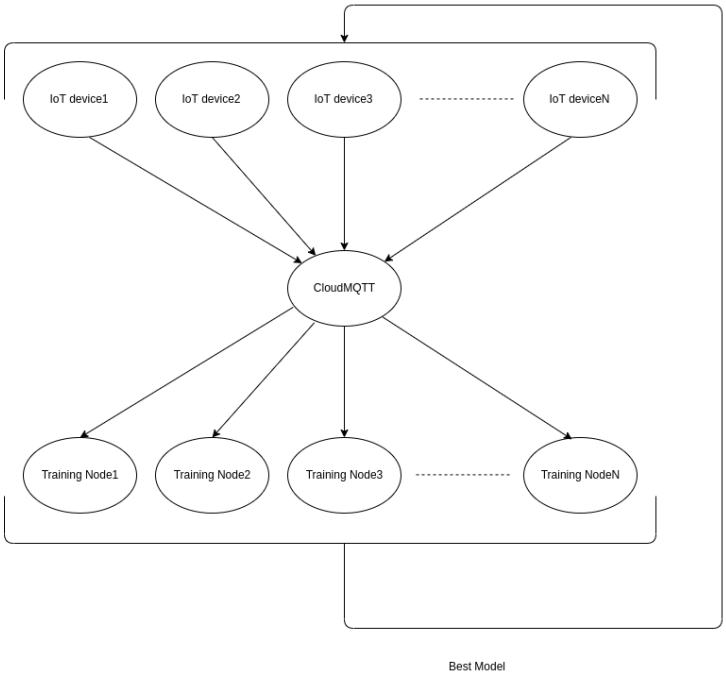


Figure 3.1: Approach Flowchart

3.3 Why Mini Batch Gradient Descent

Reasons

- Batch Gradient descent is impossible as we do not have data beforehand rather it is being sent in real time.
- Stochastic Gradient descent is not performed due to oscillations occurring in this algorithm as it uses pure online training

Because of these reasons we chose to work with Mini-Batch gradient descent which is the best of both worlds and provides high optimality in our use case.

3.4 Challenges

Challenges that we faced during the course of project:

- No access to a restricted and isolated environment to create a dataset without noise.

- Limited IoT devices to work.

We tackled these challenges by simulating the data sending of IoT devices with a static dataset. We sent data from some number of datasets, one record at a time to simulate the effect of IoT devices sending real time data.

3.5 Yield of this semester

We searched through the internet for a dataset that suits our purpose to simulate the effect of sending data by an IoT device. Once we found one, we used CloudMQTT - a globally distributed MQTT broker as our communication protocol service provider.

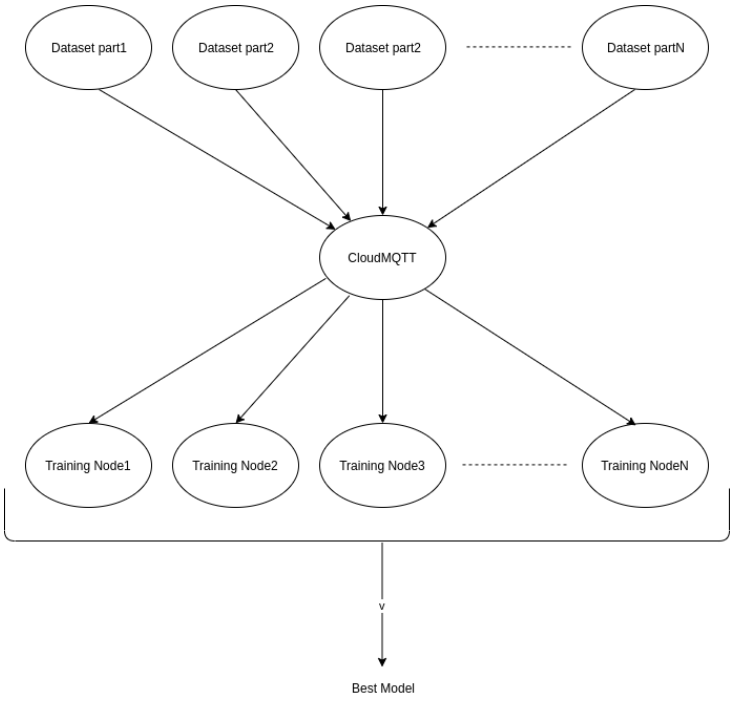


Figure 3.2: Model Diagram of the working

We sent the data in CloudMQTT and fetched the data from CloudMQTT to our training nodes. We accumulated the data in our training nodes until minimum records required are fetched then we train our model and validate it to find its accuracy. (Figure 3.2 for reference)

Chapter 4

Observations

Deciding the number of neurons in the hidden layers is a very important part of deciding your overall neural network architecture. Though these layers do not directly interact with the external environment, they have a tremendous influence on the final output. Both the number of hidden layers and the number of neurons in each of these hidden layers must be carefully considered.

Using too few neurons in the hidden layers will result in something called underfitting. Underfitting occurs when there are too few neurons in the hidden layers to adequately detect the signals in a complicated data set.

Using too many neurons in the hidden layers can result in several problems. First, too many neurons in the hidden layers may result in overfitting. Overfitting occurs when the neural network has so much information processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers. A second problem can occur even when the training data is sufficient. An inordinately large number of neurons in the hidden layers can increase the time it takes to train the network. The amount of training time can increase to the point that it is impossible to adequately train the neural network. Obviously, some compromise must be reached between too many and too few neurons in the hidden layers.

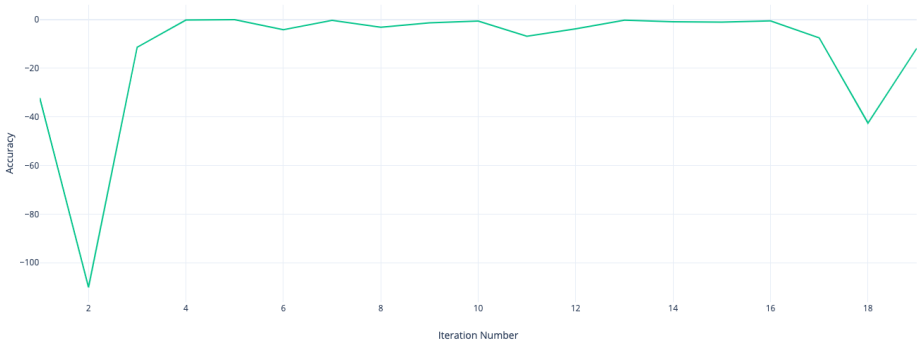
There are many rule-of-thumb methods for determining the correct number of neurons in the hidden layers, such as the following -

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be the size of the input layers, plus the size of the output layers.

- The number of hidden neurons should be less than twice the size of input layers.

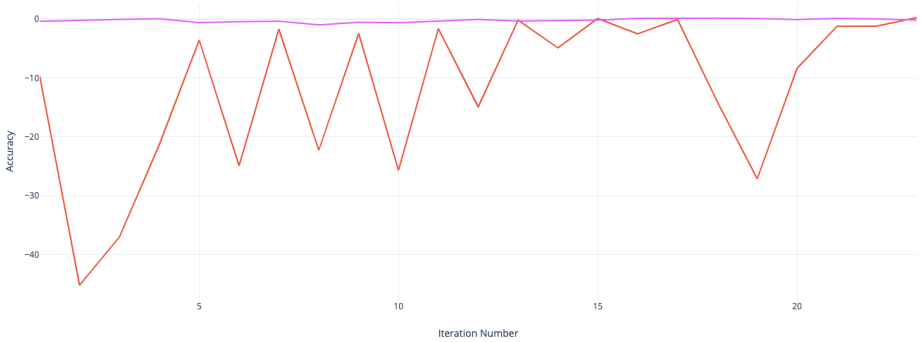
These three rules provide a starting point for you to consider. Ultimately, the selection of an architecture for your neural network will come down to trial and error. To solve this problem we created multiple nodes and trained them separately using the different real time data from datasets. We created a few models and measured their accuracy individually.

Every model had a different number of hidden layers as well as a different number of neurons in each hidden layer. Observations showed that those models which were underfitted had a very low accuracy, the ones which were overfitted had a good accuracy but their accuracy decreased gradually, the models with best-fit had a constant accuracy throughout. Let us consider an example where we consider three hidden layers with 25, 20 and 15 neurons in each layer respectively , then we get the following output which is plotted in the graph given below:

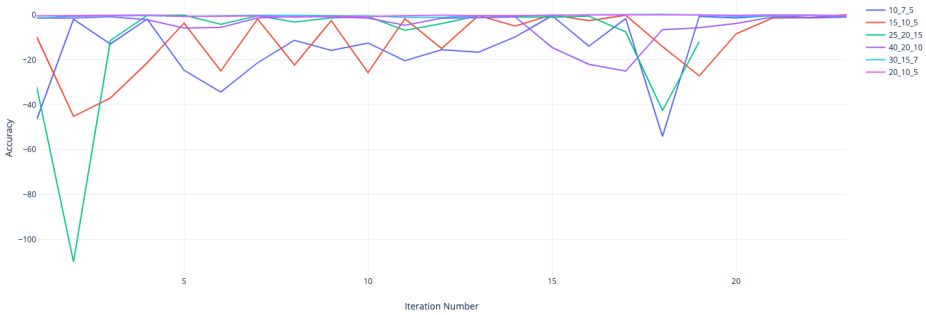


We can see from the above graph that it can be considered as a case of overfitting where the becomes constant at a point but gradually decreases with time.

In another example, a comparison has been made where we consider three hidden layers with 10, 7, and 5 neurons, and 15, 10, and 5 neurons respectively. Here we can see that the line 1 (pink line indicated hidden layer with 10, 7 and 5 neurons) can be considered more efficient as compared to the second line (red line indicated hidden layer with 15, 10 and 5 neurons) as there are very few oscillations and it produces a constant accuracy.



To find the accurate number of hidden layers and the number of neurons in each layer we tried a lot of combinations and plotted a graph based on the accuracy produced by each combination. The best possible outcome that we came to can be indicated in the following graph.



Chapter 5

Future Scope

Since this project is still in its initial stage, we have planned a number things that we will add to this project:

1. Since data is time dependent, we can implement Recurrent Neural Network(RNN)
2. We will replace the datasets with IoT devices for data sending
3. We will send the best model to all the IoT devices for them operate on their own and make predictions
4. We might use gradient descent optimization techniques such Momentum, Adagrad etc and use regularization to improve our efficiency
5. We will try to derive a mathematical function to create synthetic data and add Gaussian error to test the estimation capability.
6. Taking reference from the paper "Enabling Embedded Inference Engine with the ARM Compute Library: A Case Study"[5] we will try to optimize the architecture for faster results.

Chapter 6

Conclusion

These days machine learning techniques are widely used to solve real world problems by storing, manipulating, extracting and retrieving data from large sources. We did not store the data from the sensors but used them directly to train our models. This helped us a lot as we did not require any specific server or a definite database to store all the data.

Chapter 1 discusses all the important terms and theories related to our project in detail. We have mentioned all the techniques and terminologies, we also discussed why we have used a particular topic for our project. In Chapter 2, we discussed the details of the various tools and technologies that we have used by going through various research papers from different people. Chapter 3 explains the implementation details of the various algorithms used in various stages of our project - every detail related to the implementation has been discussed through diagrams and charts. In chapter 4, we have discussed the observations that we obtained during our implementation stages. We have also discussed the differences between choosing the hidden layers and the effects have been shown plotting graphs. In chapter 5, we discussed the future scopes and objectives of the project.

References

1. Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., & Sheth, A. P. (2018). *"Machine learning for internet of things data analysis: a survey. Digital Communications and Networks"*, 4(3), 161–175. doi:10.1016/j.dcan.2017.10.002

2. Yasumoto, K., Yamaguchi, H., & Shigeno, H. (2016). *"Survey of Real-time Processing Technologies of IoT Data Streams. Journal of Information Processing"*, 24(2), 195–202. doi:10.2197/ipsjjip.24.195

3. Earley, S. (2015). *"Analytics, Machine Learning, and the Internet of Things. IT Professional"*, 17(1), 10–13. doi:10.1109/mitp.2015.3

4. Li, H., Ota, K., & Dong, M. (2018). *""*Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. IEEE Network, 32(1), 96–101.

5. Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008). *"MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks"*. 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops

6. Dawei, Sun., Shaoshan, Liu*., & Jean-Luc, Gaudiot. (2017) *"Enabling Embedded Inference Engine with the ARM Compute Library: A Case Study"*. doi: https://arxiv.org/pdf/1704.03751.pdf

7. How to Code a Neural Network with Backpropagation In Python (from scratch) by Jason Brownlee (https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/)

8. An online guide by Michael Nielsen on Neural Networks and Deep Learning (http://neuralnetworksanddeeplearning.com/)

9. NPTEL Lecture Series on Deep Learning by Prof. Mithesh M. Khapra, IITM

10. Batch, Mini Batch Stochastic Gradient Descent by Sushant Patrikar (https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a)

11. An article about Backpropagation (https://brilliant.org/wiki/backpropagation/)

12. Documentation about CloudMQTT by CloudMQTT (https://www.cloudmqtt.com/docs/)

13. Machine Learning for Humans by Vishal Maini (https://medium.com/machine-learning-for-humans/why-machine-learning-matters-6164faf1df12)