Deep Learning on Microcontrollers

Learn how to develop embedded AI applications using TinyML

Atul Krishna Gupta

Dr. Siva Prasad Nandyala

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55518-057

www.bpbonline.com

Dedicated to

My Parents ***Mr. Keshava Kumar Gupta*** *and*
Late. Smt. Lakshmi Devi Gupta *and to*
my wife ***Richa Gupta*** *and to*
my children ***Ananya Gupta*** *and* ***Avi Gupta***

— Atul Krishna Gupta

My Parents ***Late Mr. Koti Nagaiah Nandyala*** *and*
Smt. Durgamba Nandyala *and to*
my brother ***Sambasiva Rao Nandyala*** *and to*
my wife ***Sandya Kemisetti***

— Dr. Sivaprasad Nandyala

About the Authors

- **Atul Krishna Gupta** has held many positions as Research & Development Executive in companies such as Syntiant, Macom, Inphi (now Marvell) and Gennum (now Semtech). He has over 25 years of experience in delivering all aspects of systems from IC design to software support. He has made contributions to various forums such as IEEE, SMPTE and OIF. Two technical Emmy Awards were granted to two companies for the technical work he led in the past. He was awarded with the Employee of the year award and Excellence in R&D award at Gennum. Atul holds over 20 patents. Currently, his research interests are in the field of Battery Management Systems (BMS) where he is finding ways to use AI to make Electrical Vehicles (EV) safer and last longer. He has received his B.Tech degree in Electrical Engineering from Indian Institute of Technology, Kanpur, India and MS degree in Electrical and Computer Engineering from University of Calgary, Canada.

- **Dr. Sivaprasad Nandyala** worked in Eaton Research Labs as Lead Engineer (Data Science) at Eaton India Innovation Center, Pune, India. Prior to Eaton, he worked in companies like Tata Elxsi, Wipro Technologies, Analog Devices & Ikanos Communications in multiple technology areas. Dr. Nandyala has over 35+ research publications, 1 patent grant and 6 patents under review. He obtained his Ph.D. in Speech Processing from NIT Warangal, India. He was an ERASMUS MUNDUS scholarship holder from the European government for his Postdoctoral Research at Politecnico di Milano (POLIMI), Italy.

About the Reviewer

Dr. Sanjay Boddhu is an experienced Research and Engineering leader with expertise in leading and mentoring geographically distributed teams, in the domains of Computer Vision, Image Processing, Natural Language Processing, Predictive Analytics, and Modelling. He is skilled in using various Machine Learning Ops approaches to design and develop real-world applications in Cloud and at Edge.

Acknowledgements

- **Atul Krishna Gupta:** I am incredibly grateful to Kurt Busch for providing me the opportunity to learn and contribute to the emerging field of Artificial Intelligence (AI). I would like to thank Dr. Jeremy Holleman for their insightful conversation over various topics related to neural networks. I would also like to thank Dr. Stephen Bailey for help in the firmware of the TinyML board. I would not have been able to showcase the TinyML board without guidance from Mallik Maturi and Poupak Khodabandeh.

  I would like to offer a special thanks to Zack Shelby and Aurelien Lequertier for making available their Machine Learning (ML) platform to the developer community free of cost. The platform enables zero code deployment of production grade AI deployment.

  I would like to thank my wife and children for their patience and support to finish the book.

- **Dr. Sivaprasad Nandyala:** The writing of a book is never a solo effort, and this "**Deep Learning on Microcontrollers**" book is no exception. Before anything else, I want to express my deepest gratitude to Mr. Atul Krishna Gupta who had faith in me from the beginning in writing this book. In addition, I would like to thank Dr. Sanjay Boddhu for his feedback and suggestions in reviewing the book. I want to thank the TinyML community for all the great things they have done for the field. My heartfelt thanks go to my family and friends for their unwavering support and understanding during this hard journey.

Preface

As the title of the book suggests, this book is intended to enable readers from different backgrounds to make a tangible AI application, which can be deployed on the edge on off-the-shelf platforms such as Arduino or TinyML board. The focus of this book is on the practical aspects of AI deployment. The journey of AI deployment from demo quality to production grade is not easy. We have taken a realistic example to show the pitfalls and given ideas on how to overcome the roadblocks.

While the focus of the book is on the practical side, the book also provides a good academic background as well. The field of AI is evolving and it is not practical to have one comprehensive book on all the topics, but we have given insight into some of the advanced topics of the AI field.

Deployment of AI on the edge will require some hardware. For cost effective deployment, it is expected that companies will develop their unique hardware. However, for getting started, there are several hardware boards available from websites such as Digikey or Amazon. Readers can buy this type of hardware in the range of $35-$100.

This book is divided into **9 chapters**. Each chapter description is listed as follows.

Chapter 1: Introduction to AI – will show a continuum of traditional code-based solution and Artificial Intelligence based solution. It will show where an AI based solution will be suitable and how to approach the solution.

Chapter 2: Traditional ML Lifecycle – will cover how machine learning is different from classical methods, introduction to traditional ML life cycle, performance metrics, and the basics of deep learning (DL) and different DL algorithms. It also covers transfer learning. We will discuss several tools, libraries, and frameworks for developing and deploying ML models on various embedded devices and microcontrollers. We also cover the differences between learning and inference, ML model deployment and inferencing on different hardware platforms and their comparison at various deployment levels.

Chapter 3: TinyML Hardware and Software Platforms – will cover CPUs, GPUs, Raspberry Pi boards, TPUs, and Data Center Servers. We will also look at TinyML compatible microcontrollers and Raspberry Pi boards. We then focus on TinyML's hardware boards and software platforms for machine learning. We will discuss important software platforms, data engineering, and model compression frameworks.

Chapter 4: End-to-End TinyML Deployment Phases – will discuss embedded machine learning's (EML) basics, characteristics, and examples. Next, we will also explore EML's building blocks, pros and cons, and how to run an ML model on microcontrollers. We will discuss Edge Impulse and Arduino IDE platforms, their pros and cons, and how to use different hardware boards with them. Data collection from sensors and the different platforms will be covered. We will cover data engineering, model training with Edge Impulse, optimization, and inferencing for model deployment on TinyML hardware platforms.

Chapter 5: Real World Use Cases – will cover various use cases of the TinyML deployment. The chapter categorizes these deployments in seven categories. However, many applications overlap multiple categories. These applications just show the tip of the iceberg because we just got started. Over the next few decades, we are expecting an explosion of TinyML deployment. These examples are provided just to ignite the creativity of the reader, so that they can lead innovation and deploy AI solutions which do not exist today.

Chapter 6: Practical Experiments with TinyML – will utilize Arduino IDE for TinyML hardware experimentation. We will collect sensor data using the TinyML board, clean the data for the practical experiment (Air Gesture Digit Recognition), upload it to the Edge Impulse platform, train and test the model with Nano RP2040 board sensor data. Finally, we will download the Edge Impulse inference model and test it on the RP2040 using Arduino IDE to evaluate performance.

Chapter 7: Advance Implementation with TinyML Board – will deep dive on specific hardware accelerator chips, which provide AI specific computation at a fraction of cost and power relative to microcontroller-based architecture. The development boards are readily available on these hardware accelerator chips where readers can deploy an AI solution. The power of the entire solution can run from batteries months to years. The chapter describes the entire flow of deployment in a few easy steps on the readily available Edge Impulse software platform.

Chapter 8: Continuous Improvement – will cover topics in improving the accuracy of the AI solution. AI is a data driven flow where the accuracy depends on the data. This chapter takes a deeper dive into a keyword detection application to demonstrate how to curate the data and improve the performance to take the solution from demo to production quality.

Chapter 9: Conclusion – will provide the conclusion of various aspects learned in the earlier chapters. This is an introduction book on AI and there are many topics which will require many more books. Some of those topics are mentioned in this chapter to ensure that the reader knows there is more to AI than what is covered in the book.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/yt0v6ae

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Deep-Learning-on-Microcontrollers**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Table of Contents

CHAPTER 1
Introduction to AI

Introduction

Artificial Intelligence (**AI**) today has touched all our lives without us realizing it. You may have used **Siri**, **Alexa** or **Google Assistant**. Did you ever wonder how it understands speech? During international trips, one often sees facial recognition in action, in airports. It used to take a lot of time for airline agents to check passports, but now, it is as easy as simply walking through a door. This door opens only when facial recognition is done and matches the passport. When people are sick and cannot type on their phone, all they need to do is use the voice assistant that comes along with all smartphones nowadays.

AI has become an integral part of many organizational ecosystems, not just at the consumer levels, which has brought forth many benefits such as increasing efficiency, and automating multiple tasks, while reducing installation and setup costs. For example, most machines which were installed in the last several decades, have an analog display. To replace all the monitors with digital meters is a nebulous task. Now, an image detector is placed on top of these displays, which can recognize the position of the needle and interpret the measurement in digital form. The information can be sent to the master control room via wireless protocol, such as

Wi-Fi, Bluetooth, **Long Range Radio** (**LoRa**) or even **Narrow Band IOT** (**NB-IOT**). Refer to *Figure 1.1* for an illustration of the same:

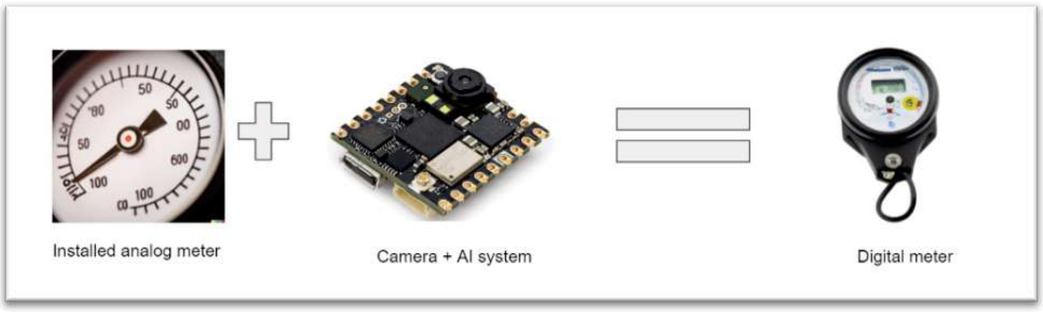


Figure 1.1: *Transforming analog to digital with AI*

Structure

In this chapter, the following topics will be covered:

- Artificial Intelligence
- Continuum of code writing and artificial intelligence
- Changing the paradigm
- Neural Network
- Machine Learning
- Intelligent IoT System vs. Cloud based IoT system
    - Arduino Nano 33 BLE Sense board
    - TinyML and Nicla Voice board
- TinyML Ecosystem
- Key applications for intelligent IoT systems

Objectives

By the end of this chapter, the reader will be able to relate what Artificial Intelligence has to offer. They will be familiar with the common lingo needed to bring an idea utilizing AI to a real system. Readers who are already doing firmware and software programming for the IoT devices can relate how their work will change when they plan to apply AI in their system. A concrete example is presented which shows where traditional methods will reach their limitations and AI deployment will be an easier path.

Artificial Intelligence

The intelligence demonstrated by computer systems is termed as artificial intelligence (Reference AI), as compared to natural intelligence demonstrated by living beings. The term **intelligence** could be controversial because as of today, the demonstrated capability of machines is still nowhere close to human intelligence. For this book, we will use the work Artificial Intelligence in the context of computers solving unique problems, which otherwise is not practical to solve with traditional code writing.

Continuum of code writing and artificial intelligence

It is expected that the reader is familiar with writing computer code. It can be argued that the problem which artificial intelligence solves, can also be solved with traditional computer code writing. However, the purpose of this book is to show that sometimes, seemingly simple problems could be very difficult to solve by traditional code writing. To appreciate the value of artificial intelligence, a hypothetical problem is posed here. Let us say an image of 200x200 pixels could contain a single line or could contain a circle. To simplify, let us assume the image is only of white or black color, as shown in *Figure 1.2*:

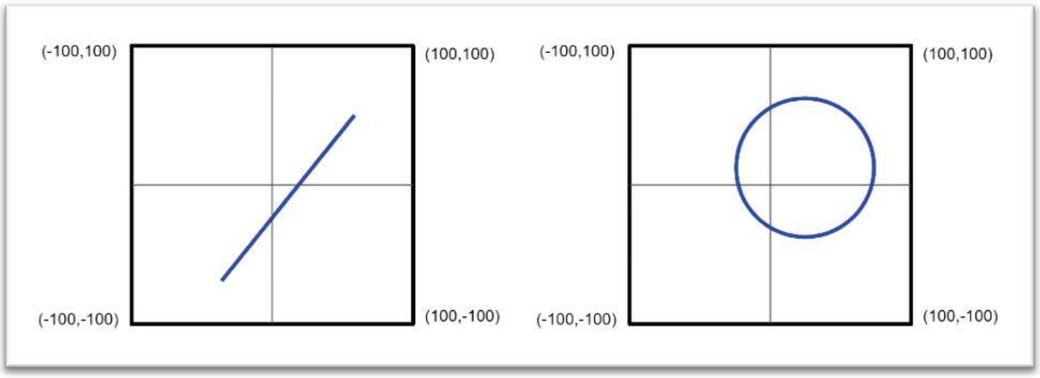


Figure 1.2: *Image containing single line or circle*

Exercise

Follow the given steps to perform the exercise:

1 Generate a 200x200 matrix with (0,0) points as origin, with value 0 (representing white space) or 1 (representing a dot in the curve).

2 Make multiple instances with lines of slope, ranging between +/-1 and y axis intercept of +/-50, as shown in *Figure 1.3*:

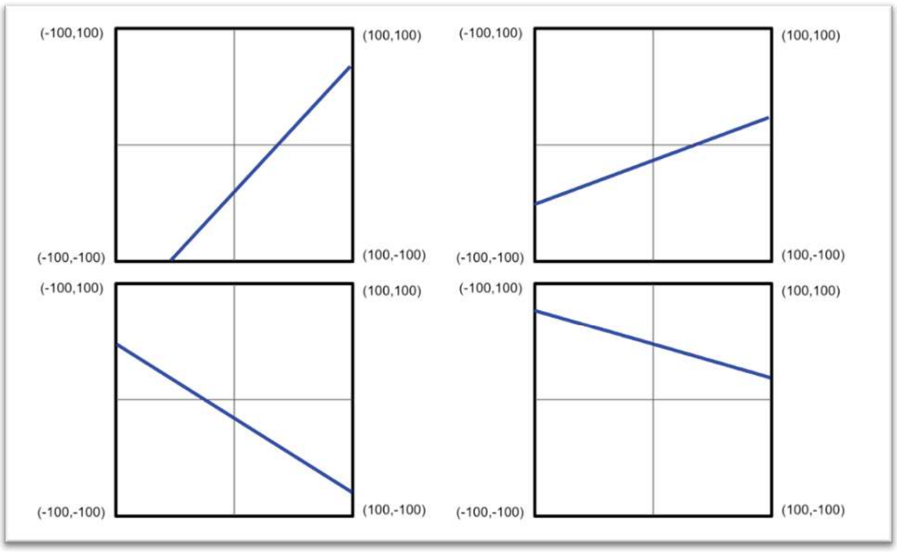


Figure 1.3: *Instances with lines of different slopes and y intercepts*

3 Similarly, make multiple instances of circles which fit completely within the image. Choose circles with radius of 10 to 100 and center within +/-50 units of (0,0) coordinate, as shown in *Figure 1.4*:

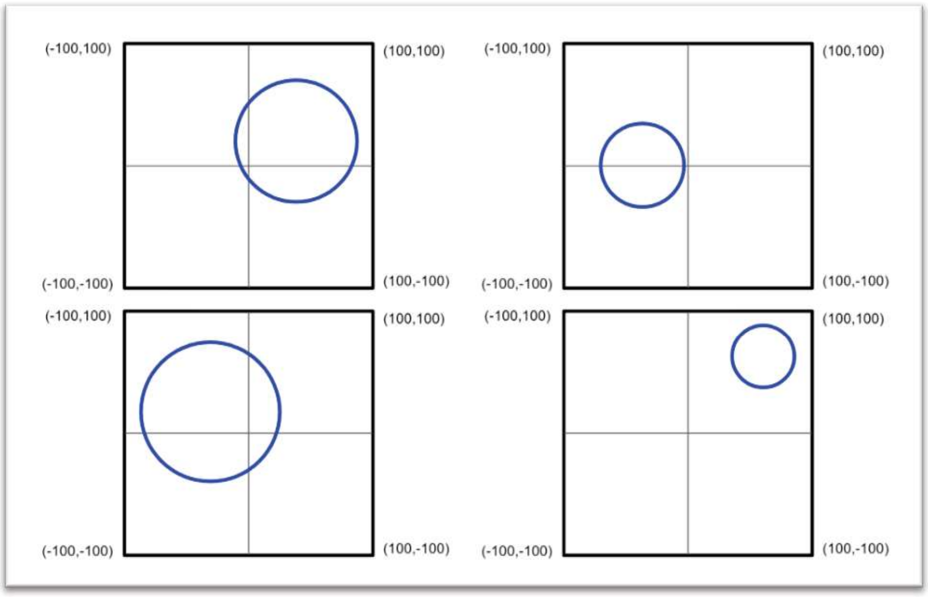


Figure 1.4: *Instances with circles*

4 Then, write a code with traditional logic and classify if this is line or circle.

5 Now extend the code to determine 0-9 digits in 28x28 pixels, using MNIST data Reference MNIST, as shown in *Figure 1.5*. If it takes more than a month to write a code to successfully recognize over 90% accuracy, then the user will appreciate the advances in artificial intelligence. The artificial intelligence methodology can find a solution with over 97% accuracy in much shorter coder's time. Please refer to the following figure:

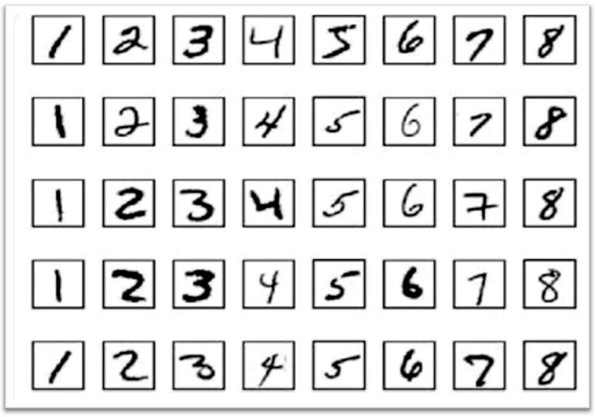


Figure 1.5: *Sample images of MNIST dataset*

In an artificial intelligence flow, the code is written once without analyzing a particular problem. The code uses already compiled libraries. Tensor flow library which is developed by Google is one such library. The user can scale the model with thousands to billions of variables which are also known as parameters. These variables are optimized during a process which is termed as a training aspect of machine learning. Thousands of data sets are required to train the system. Once the parameters are optimized, the test patterns are fed, and the classifications are checked. The test patterns are not part of the training set.

Changing the paradigm

As you may have noticed, code writing is automated at the expense of needing a lot of data for training. As the problem becomes more convoluted, it is not easy to write a traditional code even by a seasoned engineer. Writing a software which determines a face may be trivial to a seasoned engineer. However, writing a software which determines the age of the person without obvious clues, such as facial hair, is not trivial.

If the data is governed with simple laws or rules, then it is hard to justify use of artificial intelligence flow. For complex and obscure problems, such as age recognition, artificial intelligence is well suited.

Readers may be curious to know how AI programs are written. Let us consider the program which can distinguish between lines and circles, and then make it a more realistic problem where not all the points are strictly following one line or a circle. Let us consider the input image as shown in *Figure 1.6*:

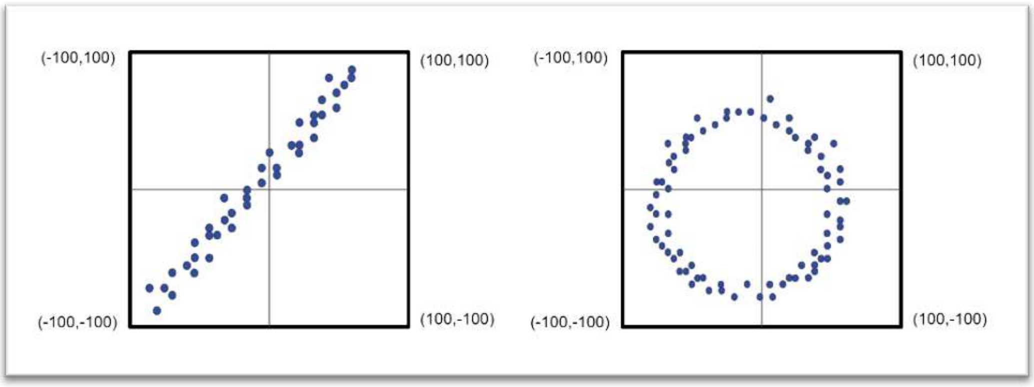


Figure 1.6: *Input image where points do not follow one line or circle*

As we know, it takes two points to define a line and three points to define a circle. Thus, we could use the same logic to define a line or circle, using specific chosen points. However, that will not be utilizing all the information, and results will also be different if different points are chosen, as shown in *Figure 1.7*:

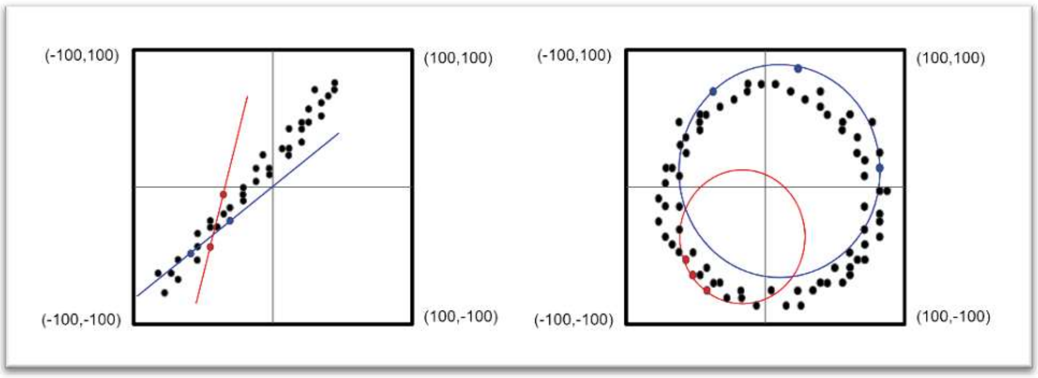


Figure 1.7: *Multiple lines or circles can be estimated if only subset of the points used*

For a robust solution, statistical regression methods should be used, which minimizes root mean square distance of all the points. All the data provided will be used, thus providing a robust solution. *Figure 1.8* illustrates the best fitting curve which is not dependent on few points but all the points:

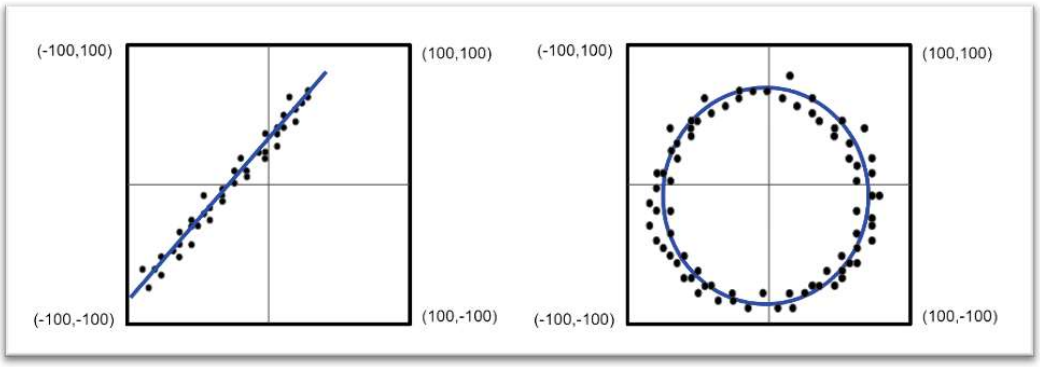


Figure 1.8: *Using regression method to find best fitted line and circle*

As you know, a line is specified as

$$Y = mX + c$$

where parameters, m and c are slope, and y intercept of the line respectively. Similarly, a circle can be written as

$$(X\text{-}Xo)\text{^}2\text{+}(Y\text{-}Yo)\text{^}2\text{=}\ r\text{^}2$$

where three parameters, *Xo,Yo* and r define the circle. *(Xo,Yo)* is the origin of the circle and r is the radius.

We can extrapolate to have several parameters to define a complex shape. The function can be defined with a set of multivariable linear equations which go through a non-linear function. We can cascade such linear and nonlinear functions to form a complex function.

To solve a generalized problem pattern recognition, the study of the biological brain has inspired a new type of processor. A biological brain contains many neuron cells which are connected to each other, making a network. It is believed that electrical signals pass through the neurons and eventually interpret a pattern. This processor is named as a neural network which indicates its origin. Let us look at the neural network and how it resembles the biological brain.

Neural Network

Most of us can guess people's age at a glance within reasonable accuracy. It comes effortlessly because this is how our brain works. Scientists got inspired from the anatomy of the biological brain, which is made of neurons. Neurons relate to multiple other neurons, which may seem a random connection. However, as the signal passes through these neurons, living beings can make rational decisions. Refer to *Figure 1.9* for an illustration of the biological neuron:

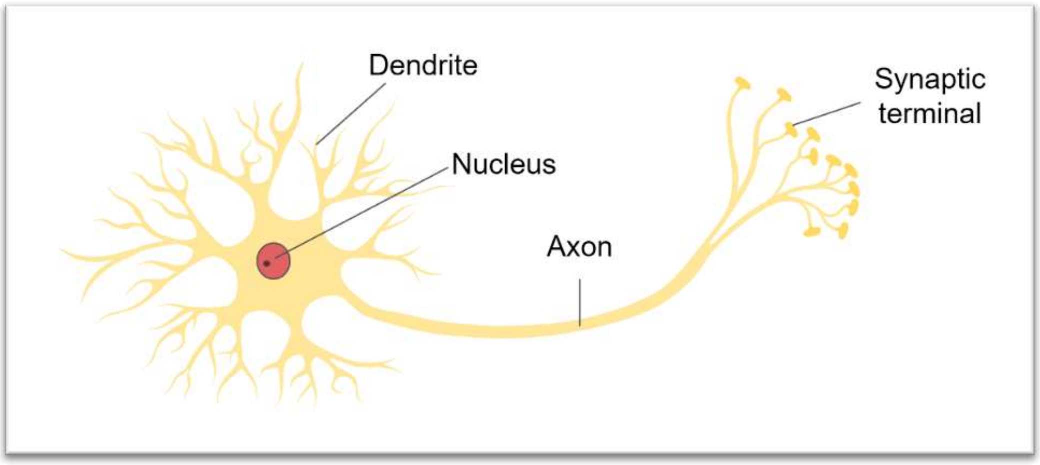


Figure 1.9: *Illustration of a biological neuron*

Figure 1.10 shows how multiple neurons are connected, thus forming a neural network. The bond between two neurons is called synaptic bond. It is believed that these synaptic bonds are being made over time. The synaptic bonds could have different connection strengths which would pass proportionate information. However, it may not be obvious how the simple connection between neurons suddenly would possess intelligence. A mathematical model made on similar principals are developed and that forms the basis of artificial intelligence. Please refer to the following figure:

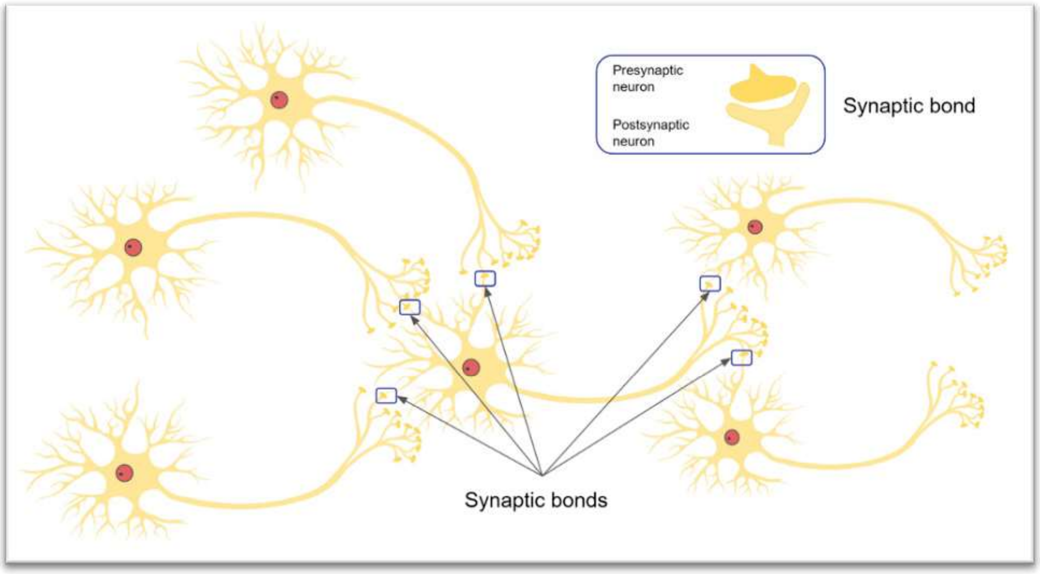


Figure 1.10: *Illustration of a biological neurons forming neural networks with synaptic bonds*

In a mathematical representation, the neuron simply sums the signals coming through the synaptic bonds and passes the signal to the next neuron. *Figure 1.11* draws a parallel between the biological neural network to a mathematical neural network. It shows how mathematical neurons are mimicking a biological neuron and how synapse connections are replaced by wights:

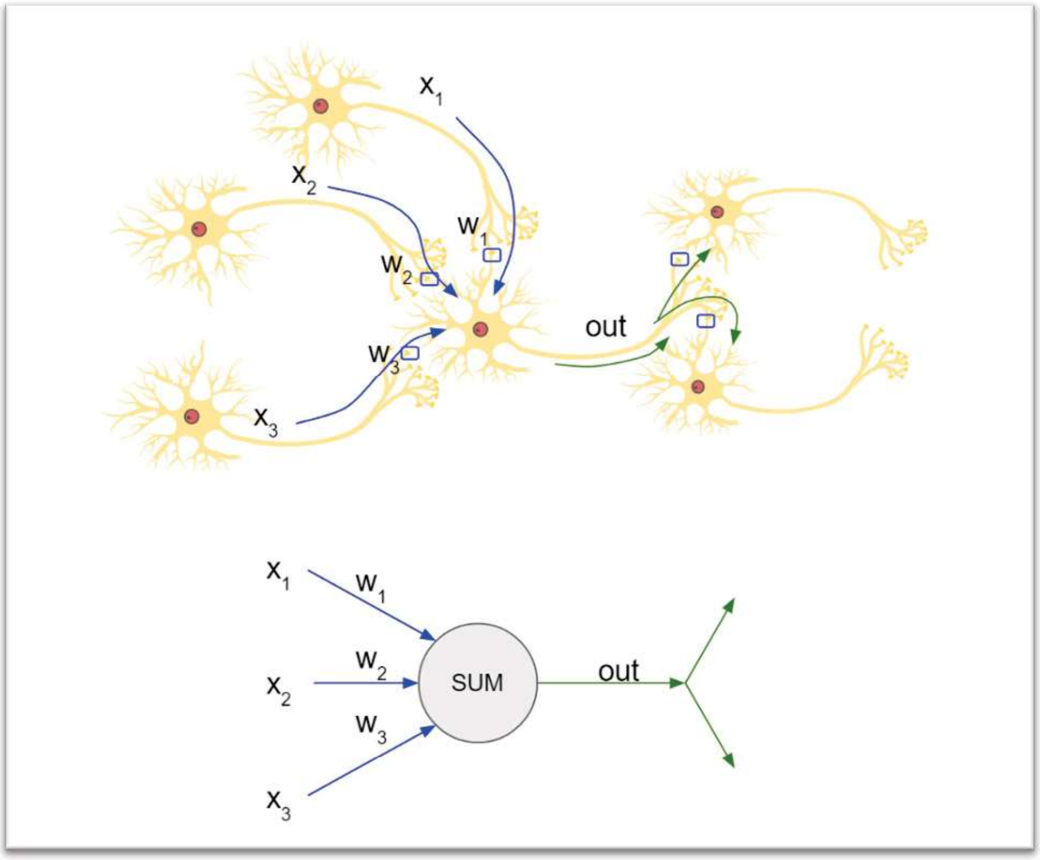
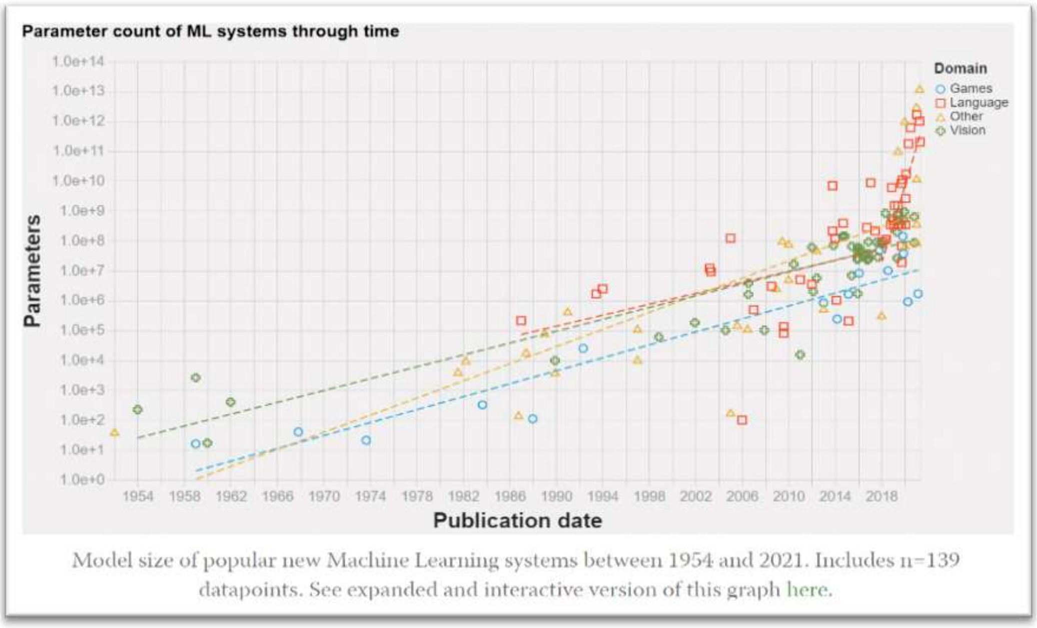


Figure 1.11: *Parallel between biological neuron and mathematical neuron*

After several tries over the years, many structured networks have been developed. The easiest network is defined as a fully connected neural network which is also termed as Dense Neural network. In a fully connected neural network, all inputs are applied to all neurons. The weight of the neuron is equivalent to the strength of the biological neuron. If there is no connection between two neurons, then the weight can be 0 in a mathematical neuron.

If one parameter is equated to one synapse of the human brain, then it is estimated that it will require several hundred trillion of parameters (reference Trillion). Let us approximate the number of parameters to be 1000 trillion parameters Assuming a typical RAM of a computer is 8 Giga Bytes, a human brain is equivalent to 1,25,000 of such laptops. A typical data center can have 1 million to 10 million servers, which can be shown to have more capacity than one human brain. So, as of today, it is not impossible to mimic the human brain in a data center. It will be a while before full emulation of human brain will be economical and widely used.

However, the number of parameters used in artificial intelligence is growing at an exponential pace. A linear-log plot shows how the number of parameters has grown since 1952 to today, as shown in *Figure 1.12*:



Model size of popular new Machine Learning systems between 1954 and 2021. Includes n=139 datapoints. See expanded and interactive version of this graph here.

Figure 1.12: *Model size of popular new Machine Learning systems between 1954 and 2021.*

Even in the linear-log curve, the plot looks exponential towards the end, showing that the growth is faster than mathematical exponential growth. As of today, the highest parameter neural model in Google search shows a 175 billion parameter model, named as OpenAI LLC's GPT-3 natural language processing model (Reference GPT3). This model still is only 1/5000 of the human brain.

It is not sufficient to just have a neural network to recognize a pattern. Even in the biological world, it takes years to train the brain. Similarly, a neural network needs to be trained. As mentioned earlier, a neural network is defined with parameters. The parameters are like variables which are placeholders for a number. For different applications, the numbers will be different. The process of finding a set of these numbers comes under machine learning. Let us take a deeper look at how machines learn.