INFOLINE

VOLUME XV ISSUE I JULY 2024

DEPARTMENT OF COMPUTER TECHNOLOGY AND INFORMATION TECHONOLOGY

KONGU ARTS AND SCIENCE COLLEGE
(Autonomous)
Affliated to Bharathiar University, Coimbatore
Accredited with A+ Grade -3.49 CGPA by NAAC
NANJANAPURAM, ERODE - 638 107

INFOLINE

EDITORIAL BOARD

EXECUTIVE COMMITTEE

STAFF ADVISOR

STAFF EDITOR

STUDENT EDITORS

CONTENTS

NEUROMORPHIC COMPUTING

Neuromorphic computing is a process in which computers are designed and engineered to mirror the structure and function of the human brain. Using artificial neurons and synapses, neuromorphic computers simulate the way our brains process information, allowing them to solve problems, recognize patterns and make decisions more quickly and efficiently than the computers that commonly use today.

The field of neuromorphic computing is still relatively new. It has very few real-world applications beyond the research being carried out by universities, governments and large tech companies like IBM and Intel Labs. Even so, neuromorphic computing shows a lot of promise particularly in areas like edge computing, autonomous vehicles, cognitive computing and other applications of artificial intelligence where speed and efficiency are imperative. Today, the scale of the largest AI computations doubles every three to four months, according to Stanford University Professor and neuromorphic computing expert Kwabena Boahen. Many experts believe that neuromorphic computing could provide a way around the limits of Moore's Law which only doubles every two years.

How Does Neuromorphic Computing Work?

Neuromorphic architectures are most often modelled after the neocortex in the brain. That's where higher cognitive functions like sensory perception, motor commands, spatial reasoning and language are thought to occur. The neocortex's layered structure and intricate connectivity are critical to its ability to process complex information and enable human thinking. The neocortex is made up of neurons and synapses that send and carry information from the brain with near-instantaneous speed and incredible efficiency. It's what tells one's foot to immediately move if one accidentally steps on a sharp nail.

Neuromorphic computers try to replicate that efficiency. They do so by forming what are called spiking neural networks. These are formed when spiking neurons, which hold data as if they were biological neurons, are connected via artificial synaptic devices that transfer electrical signals between them. A spiking neural network is essentially the hardware version of an artificial neural network, which is a series of algorithms run on a regular computer that mimics the logic of how a human brain thinks.

How Neuromorphic Computing Differs from Traditional Computing?

Neuromorphic computing architecture is a departure from the traditional computer architecture we commonly use today, which is called von Neumann architecture. Von Neumann computers process information in binary, meaning everything is either a one or a zero. And they are inherently sequential, with a clear distinction between data processing (on CPUs) and memory storage (RAM). Meanwhile, neuromorphic computers can have millions of artificial neurons and synapses processing different information simultaneously. This gives the system a lot more computational options than von Neumann computers. Neuromorphic computers integrate memory and processing more closely, too, speeding up more data-intensive tasks. Von Neumann computers have been the standard for decades, and are used for a wide range of applications, from word processing to scientific simulations. But they're energy inefficient and often run into data transfer bottlenecks that slow down performance. And as time goes on, von Neumann architectures will make it increasingly more difficult to deliver increases in compute power that we need. This has led researchers to pursue alternative architectures like neuromorphic and quantum.

Neuromorphic Computing vs. Quantum Computing

Neuromorphic computing and quantum computing are two emerging approaches to computation, each with its own distinct set of characteristics, advantages and applications.

Neuromorphic computing

- It is inspired by the structure and functionality of the human brain.
- It uses artificial neurons and synapses to accomplish parallel processing and real-time learning.
- It is well suited for tasks involving pattern recognition and sensory processing.
- It is logistically easier to accomplish than quantum computing.
- It is more energy efficient than quantum computing.

Quantum computing

- It leverages principles of quantum mechanics to process information.
- It relies on qubits (quantum bits) to run and solve multidimensional quantum algorithms. It is especially good at efficiently solving complex problems like cryptography and molecular simulation.
- It requires lower temperatures and uses more power than neuromorphic computers.

Although they are quite different from one another, both neuromorphic and quantum computing hold significant promise in their

own rights, and are still very much in the early stages of development and application.

Benefits of Neuromorphic Computing

Faster Than Traditional Computing

Neuromorphic systems are designed to imitate the electrical properties of real neurons more closely which could speed up computation and use less energy. Because of they are operating in an event-driven way, where neurons only process information when relevant events occur, they can generate responses. Low latency is always beneficial, but it can make a big difference in tech that relies on real-time sensor data processing, like IoT devices.

Excellent at Pattern Recognition

Because neuromorphic computers process information in such a massively parallel way, they are particularly good at recognizing patterns. By extension, this means they're also good at detecting anomalies, Accenture Labs' Danielescu said, which can be useful in anything from cybersecurity to health monitoring.

Able to Learn Quickly

Neuromorphic computers are also designed to learn in real-time and adapt to changing stimuli, just as humans can, by modifying the strength of the connections between neurons in response to experiences. This versatility can be valuable in applications that require continuous learning and quick decision-making, whether that's teaching a robot to function on an assembly line or having cars navigate a busy city street autonomously.

Energy Efficient

One of the most prominent advantages of neuromorphic computing is its energy efficiency, which could be especially beneficial in the making of artificial intelligence a notoriously wasteful industry. Neuromorphic computers can process and store data together on each individual neuron, as opposed to having separate areas for each the way von Neumann architectures do. This parallel processing allows multiple tasks to be performed simultaneously, which can lead to faster task completion and lower energy consumption. And spiking neural networks only compute in response to spikes, meaning only a small portion of a system's neurons use power at any given time while the rest remain idle.

Neuromorphic Computing Uses

Despite these challenges, neuromorphic computing is still a highly funded field projected to be worth some $8 billion, according to one report. And experts are enthusiastic about its potential to revolutionize various tech fields thanks to its unique ability to mimic the brain's information processing and learning capabilities.

Self-Driving Cars

Self-driving cars must make instant decisions to properly navigate and avoid collisions, which can require extensive

computing power. By employing neuromorphic hardware and software, self-driving cars could be able to carry out tasks faster than if they used traditional computing, all with lower energy consumption. This can make for quicker response times and corrections on the road while also keeping overall energy emissions down.

Drones

Using neuromorphic computing, drones could be just as responsive and reactive to aerial stimuli as a living creature. This technology may allow vision-based drones to autonomously traverse complex terrain or evade obstacles. A neuromorphic-engineered drone can also be programmed to only increase its energy usage when processing environmental changes, allowing it to rapidly respond to sudden crises such as in rescue or military operations.

Edge AI

Neuromorphic computing's energy efficiency, adaptability and ability to process data in real-time make it well-suited for edge AI, where computations are done locally on a machine (like a smart device or autonomous vehicle) rather than in a centralized cloud computing facility or offsite data center, requiring the real-time processing of data from things like sensors and cameras. With its event-driven and parallel-processing capabilities, neuromorphic computing can enable quick, low-latency decision-making and its energy efficiency can extend the battery life of these devices, reducing the need to recharge or replace edge devices around the home. In fact, Bron said some studies have found neuromorphic computing to be 100-times more effective in terms of battery efficiency than normal computing.

Robotics

Neuromorphic systems can enhance the sensory perception and decision-making capabilities of robots, enabling them to better navigate complex environments (like a factory floor), recognize objects and interact with humans more naturally.

Fraud Detection

Neuromorphic computing excels at recognizing complex patterns and could therefore identify subtle patterns indicative of fraudulent activity or security breaches such as unusual spending behaviour, unauthorized or counterfeit login attempts. Plus, the low latency processing of neuromorphic computing could enable a swifter response once the fraud has been detected such as freezing accounts or alerting the proper authorities in real time.

Neuroscience Research

Through its use of brain-inspired neural networks, neuromorphic computing hardware is used to advance our understanding of human cognition. As researchers try to recreate our thought processes in electronics, they may learn more about the brain's inner workings. In 2020, Intel partnered with Cornell University to essentially teach its neuromorphic computer

chip Loihi how to identify smells. Eventually the researchers said they would like to extend their approach to processes like sensory scene analysis and decision-making, helping them to understand how the brain's neural circuits solve complex computational problems. The Human Brain Project, an EU-funded group made up of some 140 universities, teaching hospitals and research centers, spent ten years attempting to create a human brain using two neuromorphic supercomputers. It concluded its work in September of 2023.

S.Dinesh

III B.Sc. (Computer Technology)

PostgreSQL

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 35 years of active development on the core platform. PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001 and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organisations.

PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help one manage one's data no matter how big or small the dataset. In addition to being free and open source, PostgreSQL is highly extensible. PostgreSQL tries to conform with the SQL standard where such conformance does not contradict traditional features or could lead to poor architectural decisions. Many of the features required by the SQL standard are supported, though sometimes with slightly differing syntax or function. Further moves towards conformance can be expected over time. As of the version 16 release in September 2023, PostgreSQL conforms to at least 170 of the 179 mandatory features for SQL:2023 Core conformance.

Features of PostgreSQL

Data Types

- Primitives: Integer, Numeric, String, Boolean
- Structured: Date/Time, Array, Range / Multirange, UUID

- Document: JSON/JSONB, XML, Key-value (Hstore)
- Geometry: Point, Line, Circle, Polygon
- Customizations: Composite, Custom Types

Data Integrity

- UNIQUE, NOT NULL
- Primary Keys
- Foreign Keys
- Exclusion Constraints
- Explicit Locks, Advisory Locks

Concurrency, Performance

- Indexing: B-tree, Multicolumn, Expressions, Partial
- Advanced Indexing: GiST, SP-Gist, KNN Gist, GIN, BRIN, Covering indexes, Bloom filters
- Sophisticated query planner / optimizer, index-only scans, multicolumn statistics
- Transactions, Nested Transactions (via savepoints)
- Multi-Version concurrency Control (MVCC)
- Parallelization of read queries and building B-tree indexes
- Table partitioning
- All transaction isolation levels defined in the SQL standard, including Serializable
- Just-in-time (JIT) compilation of expressions

Reliability, Disaster Recovery

- Write-ahead Logging (WAL)
- Replication: Asynchronous, Synchronous, Logical
- Point-in-time-recovery (PITR), active standbys
- Tablespaces

Security

- Authentication: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificate and more
- Robust access-control system
- Column and row-level security
- Multi-factor authentication with certificates and an additional method

Extensibility

- Stored functions and procedures
- Procedural Languages: PL/pgSQL, Perl, Python, and Tcl. There are other languages available through extensions, e.g. Java, JavaScript (V8), R, Lua, and Rust
- SQL/JSON constructors and path expressions
- Foreign data wrappers: connect to other databases or streams with a standard SQL interface
- Customizable storage interface for tables
- Many extensions that provide additional functionality, including PostGIS

Internationalisation, Text Search

- Support for international character sets, e.g. through ICU collations
- Case-insensitive and accent-insensitive collations
- Full-text search

There are many more features that one can discover in the PostgreSQL documentation. Additionally, PostgreSQL is highly extensible: many features such as indexes, have defined APIs so that one can build out with PostgreSQL to solve one's challenges. PostgreSQL has been proven to be highly scalable both in the sheer quantity of data it can manage and in the number of concurrent users it can accommodate. There are active PostgreSQL clusters in production environments that manage many terabytes of data and specialized systems that manage petabytes.

S.Dharshini

I B.Sc. (Information Technology)

AUTOMATED TECHNIQUE FOR ANIME COLORIZATION USING DEEP LEARNING

Researchers report the world's first technique for automatic colorization focused on Japanese anime production. The new technique is expected to promote efficiency and automation in anime production. Japanese researchers from IMAGICA GROUP Inc., OLM Digital, Inc. and Nara Institute of Science and Technology (NAIST) have jointly developed a technique for automatic colorization in anime production.

While the number of animation works produced in Japan has been increasing every year, the number of animators has remained almost unchanged. To promote efficiency and automation in anime production, the research team focused on the possibility of automating the colorization of trace images in the finishing process of anime production. By integrating the anime production technology and know-how of IMAGICA GROUP Inc. and OLM Digital, Inc. with the machine learning, computer graphics and vision technology of NAIST, the research team succeeded in developing the world's first technique for automatic colorization of Japanese anime production. The technique is based on recent advances of deep learning approaches that are nowadays widely applied in various fields.

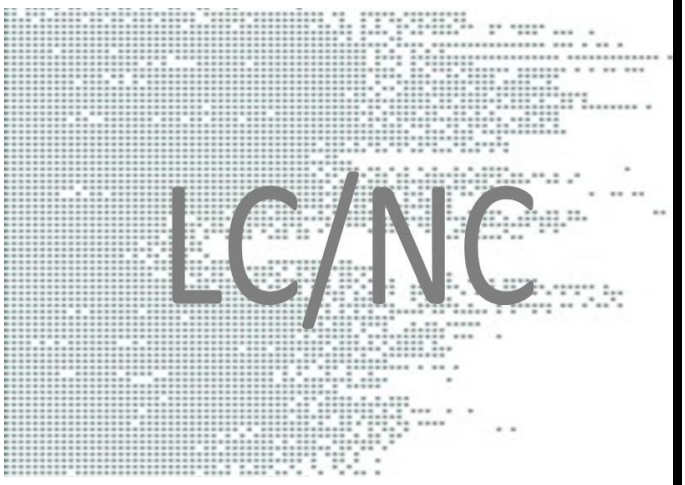
After the trace image cleaning in a pre-processing step, automatic colorization is

performed according to the color script of the character using a deep learning-based image segmentation algorithm. The colorization result is refined in a post-process step using voting techniques for each closed region.

A.P.Anbu

I B.Sc. (Information Technology)

UNDERSTANDING LOW-CODE NO-CODE (LCNC) PLATFORMS



The IT-enabling units of most organizations face tremendous pressure to develop and deploy software applications rapidly as per end-user expectations. New software developments are often put on hold or delayed simply because of shortage of skilled resources and a backlog of operations and maintenance work. Developers of all hues are expected to provide more applications in less time than ever before, and because developer talent is in short supply, organizations must equip their existing developers with tools and platforms. When compared to traditional approaches, low-code platforms simplify, speed and lower the cost of application development, which is particularly tempting to the IT functions.

The digital workplace is adopting low-code and no-code applications as their preferred technology. The focus will shift from application development to assembly and integration. According to Gartner's recent research, by 2025, 70% of new apps built by enterprises would use low-code or no-code technologies, up from less than 25% in 2020. While software is developed to address business problems, internal IT teams are losing ground to business teams who can no longer wait months for a server to be provisioned when they can simply go to the cloud with a credit card. Similarly, they are experimenting with new applications on their own, sometimes without the involvement of IT. Business teams/Client can design their own applications using a low-code/no-code approach. Because of the reduced cost and lower technical barriers to entry, many businesses are prioritizing the digital transformation of daily operations and procedures using low-code/no-code tools. Professional developers may perform jobs two to three times faster with these technologies than with typical developer

tools. Low-code technologies also allow business users and developers to focus on their area of expertise thereby reducing friction.

Advent of LCNC

Low-code and no-code development platforms are tools for people who either do not know how to code or have no time to code. Whereas these low-code and no-code frameworks are built on actual coding languages like PHP, Python, and Java, end users are not concerned with the specifics. Instead, they are given visual software development environments where they can drag and drop program components, link them and watch what occurs. In effect, it may be utilized as a familiar wizard-style paradigm to build, test and even deploy apps that are totally focused on simplicity of use.

Many people consider Visual Basic as one of the earliest low-code integrated development environment (IDE). However, the first true low-code application platforms arrived in the late 1990s and early 2000s as fourth-generation programming languages and fast application development platforms. Another approach to LCNC platforms began with the spreadsheet, which has a long history dating back to the 1960s. It is a non-procedural, non-algorithmic approach to computation that became immensely popular. Spreadsheets have enabled a whole generation of businesses to efficiently use computers without knowledge of programming.

It is interesting to take note of the evolution of programming itself. Assembler is low-code in comparison to machine language and toggling switches for directly inserting binary instructions into the computer's memory. C and FORTRAN are low-code in comparison to assembler; Python is low-code in comparison to C++. Rather than writing everything from scratch, developers can rely on the Python runtime environment and libraries, which contain millions of lines of code.

Advantages of LCNC

LCNC promises several advantages such as:

(i) Faster Development and Launch Cycle: Low-code development speeds up enterprise application development. The web-based drag-and-drop functionalities, as well as reusable application components and in-built libraries aid in the application design process. Organizations can launch their applications to faster and make changes on short notice as a result of this.

(ii) Degree of Freedom: LCNC platforms may differ based on degree of freedom to configure offerings by them. The higher number of parameters offered to configure on the fly, in any platform business scope, provides better chance to end-users to model their requirements

(iii) Tenancy Extension: While it is possible to make a LCNC platform for most of the requirement, still there is a possibility that some peculiar needs remain unaddressed by the platform framework and hence platform must

support provision of tenancy extension in easier way using some scripting language to tweak the behaviour of LCNC platform to meet peculiar needs.

(iv) Promotes Innovation: Developers can use LCNC to demonstrate their ideas quickly and can be ramped up during implementation. Instead of simply providing theory, a developer can demonstrate how the project would shape up in order to gain executive buy-in and persuade management to devote more development resources to the initiative. The innovation using LCNC is further compounded with Degree of Freedom and Tenancy Extension features.

(v) Customer Experience: Customers who grew up with digital devices expect similar experiences when using any consumer apps. Low code allows easy integration with numerous services more quickly, maintaining a uniform Omni channel experience.

(vi) Lower IT Infrastructure needs: Most LCNC applications are available or deployable on the cloud, providing on-demand scalability and drastically reducing the upfront investment needs in IT infrastructure. LCNC ensures faster innovation in less time while reducing IT staffing.

(vii) Increases Efficiency: No-code and low-code apps are good for enhancing everyday task efficiency because they need less development effort. IT units can address their own operational difficulties without needing many coders if they can build their own versions of familiar apps or swiftly install much-needed functionalities.

(viii) Efficient Governance: A LCNC platform allows IT and DevOps teams to manage more efficiently a portfolio of applications while maintaining complete compliance and governance capabilities. Low code decreases reliance on third-party apps for quick fixes and allows for a collaborative work environment.

(ix) Easy to understand: Traditional codes are often difficult to understand as they are written by several developers and often is not easy to read and interpret. Debugging of such code also is very time consuming. Low-code and no-code platforms are easier to understand and thus, easier to identify configuration/script bugs and fix them.

(x) Increased Agility: In a rapidly evolving digital world, businesses need to adapt and respond effectively to changing scenarios and dynamics along with leveraging potential opportunities. Using a traditional development approach to create applications is too time-consuming. LCNC development tools help speed up deployment of an almost endless set of requirements more rapidly with little risk.

Disadvantages of LCNC

On the flip side, LCNC platforms have a few disadvantages as well:

(i) Reduced flexibility: With traditional codes, developers can customize their software as per user requirements, however LCNC solutions and plugins have built-in functionality and is

not always easy to fulfil related requirements. Though this can be largely overcome with provisions of Tenancy extensions using simple scripting language.

(ii) Security and risk: LCNC platforms rely heavily on their platform providers to mitigate IT risks and security flaws as the application providers do not have any control over the source code. If these platform providers discontinue services, no security updates would be available, and applications will be unable to fix them. Moreover, businesses relying on LCNC providers risk their data and systems being exposed and vulnerable to security breaches. This is applicable for LCNC products acquired from the vendors and not applicable for in-house developed products.

(iii) Vendor lock-in: Applications using a specific LCNC platform for their IT solution, makes it difficult for them to switch to a different platform. This increases the dependency of the business on an individual LCNC provider. Obviously, this is not applicable for in-house developed products

An LCNC solution would generally limit customization options to a large extent which can be only done with enterprise wide approach during architecture phase) within the domain of discourse (Business Boundary). Generic applications are often easy fit and more adaptable to LCNC solutions, while others are not. Before adopting any platform, customization requirements need to be carefully assessed. Ultimately, the decision to use a LCNC platform depends on each business's objectives and needs.

LCNC Architectural Guiding principles

The core guiding principles of architecting a LCNC Platform is that a) architect must have in-depth knowledge of domain of discourse (or enterprise wide), b) Presence of Enterprise view while architecting In order to capture diverging requirements and generalize them c) must possess strong fundamentals of journey from Generalization to specialization approach & Vice versa and d) the architect must be committed to avoid any possibility of "hard-coding" the business logic or process or layouts in any way and c) the proposed architecture should be open to evolution.

The simplest & indicative yardstick to determine their proximity to LCNC architectural principle is [(Number of parameters (Variables) available to client to configure on the fly (without writing code) / Number of total parameters(Variables) available in the platform) * 100]. Higher the percentage value of it would indicate better degree of configurability meaning better LCNC Platform. Obviously, a better LCNC Platform would have no hardcoded parameters within the boundary of platform business scope.

M.Harini

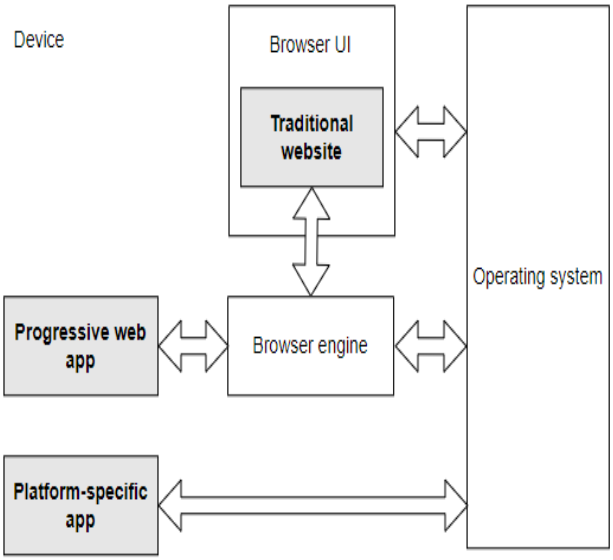
III B.Sc. (Computer Technology)

PROGRESSIVE WEB APP

A progressive web application (PWA), or progressive web app is a type of application software delivered through the web, built using common web technologies including HTML, CSS, JavaScript and Web Assembly. It is intended to work on any platform with a standards-compliant browser, including desktop and mobile devices. Since a progressive web app is a type of webpage or website known as a web application, it does not require separate bundling or distribution. Developers can simply publish the web application online, ensure that it meets baseline installation requirements and ensure that users will be able to add the application to their home screen.

When one visit a website in the browser, it's visually apparent that the website is "running in the browser". The browser UI provides a visible frame around the website, including UI features like back/forward buttons and a title for the page. The Web APIs one's website calls are implemented by the browser engine.

PWAs typically look like platform-specific apps they are usually displayed without the browser UI around them but as a matter of technology, still websites. This means they need a browser engine, like the ones in Chrome or Firefox, to manage and run them. With a platform-specific app, the platform OS manages the app, providing the environment in which it runs. With a PWA, a browser engine performs this background role, just like it does for normal websites.



The browser starts a PWA's service worker when a push notification is received. Here, the browser's activity is entirely in the background. From the PWA's point of view, it might as well be the operating system that started it. For some systems, such as Chromebooks, there may not even be a distinction between "the browser" and "the operating system."

Advantages of PWA over traditional web and native apps

- The progressive web apps can work on any device with a browser. As a result, they eliminate the need for separate apps for different platforms.
- These apps work offline or with poor connectivity. As a result, they are providing a seamless user experience.

- The progressive web apps are faster and more responsive than traditional ones. Therefore, leading to higher user engagement.
- Users can access them directly from the web without downloading from an app store.

K.Bharathkumar

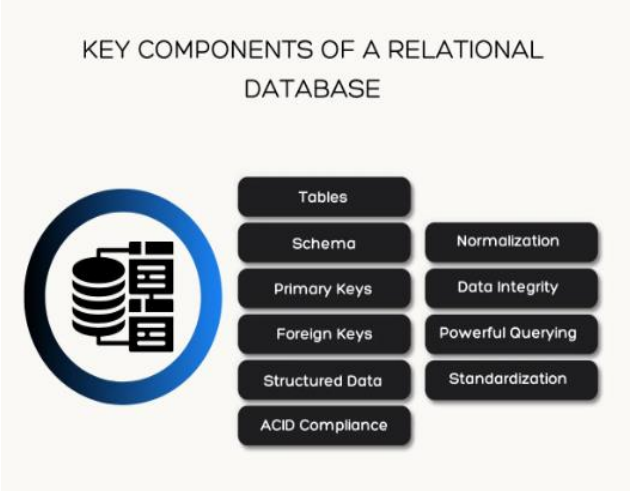
III B.Sc. (Information Technology)

RELATIONAL DATABASE VS NON-RELATIONAL DATABASE

The core difference between a relational database and a non-relational database is how data is structured and organized within each type. Software applications can vary significantly in how they handle data storage, access, and management based on the type used. The choice significantly impacts an application's performance, scalability and complexity.

Relational Database (SQL)

A relational database or a SQL (structured Query Language) database organizes data into tables (or relations) containing rows and columns. Each table represents a specific entity, while keys define the relationships between the tables. These keys allow for complex querying and data manipulation when necessary.



Tables: The primary structure where data is stored. Each table contains rows and columns, with rows representing records and columns representing fields in a record. **Schema:** Defines the structure of the database, including tables, columns, data types and relationships between tables.

Primary Keys: Unique identifiers for each record in a table, ensuring that each record can be uniquely identified.

Foreign Keys: Fields that create a link between two tables, establishing relationships between them.

The components of a relational database allow for unique characteristics that enable users to ensure data consistency and integrity throughout the software application. These characteristics are:

- **Structured Data:** Data is stored in a highly structured format, making it easy to query and manipulate using SQL.
- **ACID Compliance:** Ensures reliable transactions through Atomicity,

Consistency, Isolation and Durability properties.

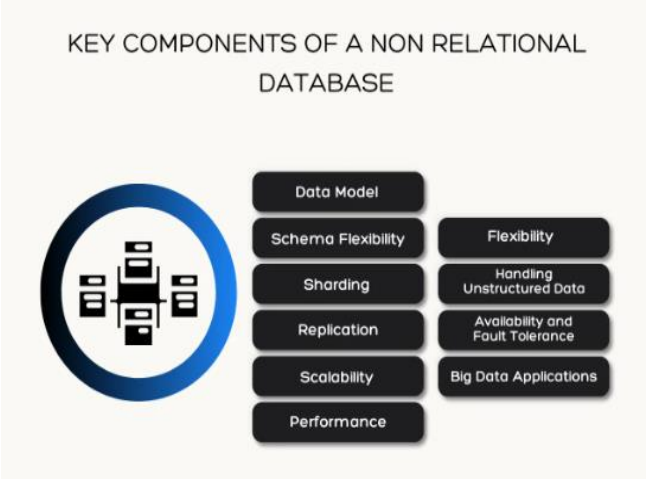
- **Normalization:** Organizing data to minimize redundancy and improve data integrity.

With its structured characteristics, a relational database provides software developers with these main advantages:

- **Data Integrity:** StroStrong enforcement of data integrity and relationships, reducing the likelihood of data anomalies.
- **Powerful Querying:** Robust SQL querying capabilities, enabling complex joins, filters and aggregations.
- **Standardization:** SQL is widely accepted, ensuring compatibility and interoperability across different systems.

Non-Relational Database (NoSQL)

A non-relational database or NoSQL (Not Only SQL) database, provides a way to store and retrieve data that does not adhere strictly to the tabular design schema of relational databases. NoSQL databases are better suited for handling large volumes of varied data types and sizes while keeping consistent metrics for high-performance operations, making them ideal for big data and real-time web applications.



Data Model: Large variety of data types from document-based, key-value pairs, column-family, or graph-based structures.

Schema Flexibility: No fixed schema; data models can be dynamically adjusted to fit based on the design and needs of the application.

Sharding: Data is horizontally partitioned into multiple servers to ensure scalability options remain viable for long-term usage.

Replication: Real-time web applications that deal with large and repetitive data benefit from NoSQL databases' ability to replicate data across multiple nodes.

Each component offers a more extensive selection of options based on the developer's specifications and application requirements. High-performance operations are more accessible to track and manage under the right database practices. NoSQL platforms heavily rely on these characteristics to stand out:

Scalability: Horizontal scaling makes distributing this non-relational database through multiple servers easier.

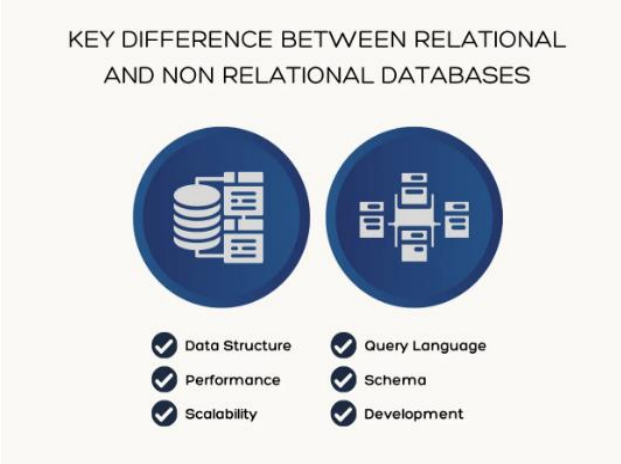
Performance: High read/write performance suits real-time applications and programs perfectly.

Flexibility: Schema less design allows for easy adaptation to changing data requirements. These characteristics provide various advantages that set it apart from the other types of databases. These benefits are:

- **Handling Unstructured Data:** Semi-structured and unstructured data storage and manipulation are easier to handle while also opening for complex systems for management.
- **Availability and Fault Tolerance:** Often designed with built-in replication and distribution features, ensuring high availability.
- **Big Data Applications:** Suitable for handling large volumes of data generated at high velocity.

Differences Between Relational and Non-Relational Databases

With a high-level understanding of relational vs. non-relational databases, one can now compare them against each other when it comes to their key differences. To fully understand these differences, it's important to look at how they differ regarding data structure, performance, scalability, schema and development.



1. Data Structure

A data structure is how data is organized and stored in a database. Relational databases use a table-based structure, while non-relational databases use different data models such as key-value, document, column families and graphs.

A relational database is well-suited for any application that demands a high level of data organization, integrity and relational mapping. On the other hand, software applications that require flexible, scalable solutions to handle large volumes of diverse and rapidly changing data are better suited for non-relational databases.

2. Performance

When it comes to performance between a relational database vs non-relational database, a relational database provides strong data consistency and integrity. A non-relational database performs faster for specific use cases like big data and real-time processing.

3. Scalability

Scalability in terms of relational and non-relational databases refers to the ability of the database to handle increasing amounts of data and user load without compromising performance. It involves the database's capacity to expand its resources and capabilities to accommodate growth.

Relational databases have limited scalability, making them less suitable for large datasets and high read/write loads. Non-relational databases are highly scalable and can more efficiently handle large-scale, distributed data.

4. Query Language

A query language is a specialized language used to communicate with a database to retrieve, manipulate and manage data. It allows users and applications to interact with the database by specifying queries that describe the data they want to access or manipulate.

Relational databases use SQL for querying and manipulating data. In contrast, non-relational databases typically use query languages or APIs which can vary between databases.

5. Schema

A schema is a structure or blueprint that defines how data is organized and represented in a database. It defines the tables, fields, relationships and constraints that ensure data integrity and consistency within the database.

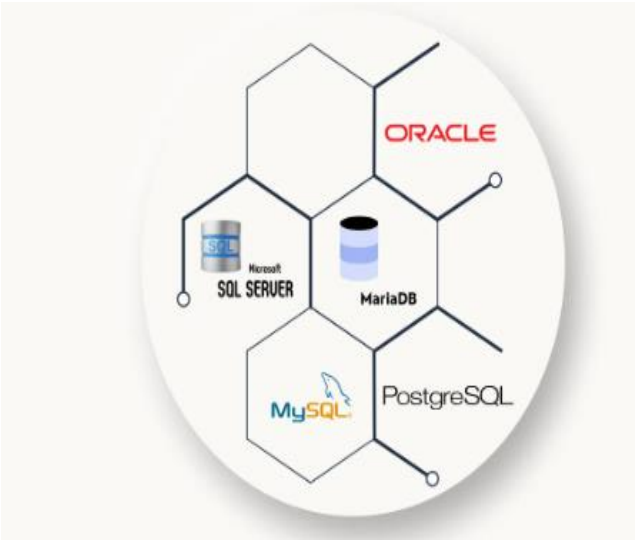
Relational databases have a predefined schema, making them better suited for structured data. Non-relational databases, however, are more flexible and can accommodate various types of data.

6. Development

In terms of development and a relational database vs a non-relational database, relational databases require more development effort when creating complex queries or changing the database structure. Non-relational databases are easier to develop and require fewer resources.

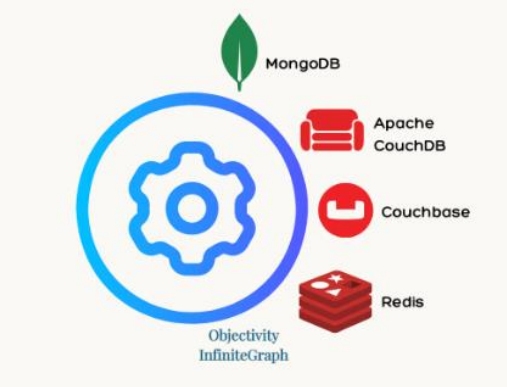
Popular Relational Databases (SQL Databases)

Popular relational databases used in software development include open-source and commercial options that cater to various needs, from small-scale applications to large enterprise systems. Here are some of the most widely used relational databases:



Popular Non-Relational Databases (No SQL Databases)

Non-relational databases, or NoSQL databases, are designed to handle large volumes of diverse data types and provide high performance and scalability. Here are some of the most widely used NoSQL databases:



M.Harini

II B.Sc. (Computer Technology)

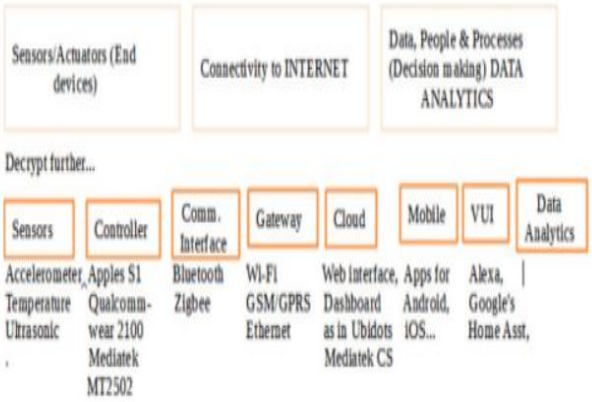
METHODOLOGIES IN INTERNET OF THINGS (IOT)

The Internet of Things (IoT) is an emerging technology with a lot of applications and a potential area of research. Low power Wireless Personal Area Networks(6LoWPANs) are IPv6 enabled which participate an essential role in IoT. IPv6 is suited to Internet integration, low power energy consumption and wireless ubiquitous availability. The aim of IoT is to create a favourable situation for devices from anywhere to communicate with each other. The term Internet of Things was first begot by Kevin Ashton (1999) which prompted mechanical upheaval for the fate of registering and correspondences. Its advancement relies upon a dynamic specialized development in varied vital fields, commencing wireless sensors to the nano-technology based architectures. This type of technology includes smart city, control activation and support of complex frameworks in industry field, well-being, communication and transport.

IoT Components

The components of IoT are:

- Device(The Thing)
- Local network
- Internet
- Backend services
- Applications



Methodologies for IoT

Simulators

There are two versions of Network Simulators (NetSim). They are Network Simulator Version 2 (NS2) and Network Simulator Version 3(NS3). A NS2 replication tool is an occasion driven which is extensively used in studying and considerate the active environment of the announcement network. The other simulators are: IOTify, ifog simulator, Cooja simulator,

MATLAB, CISCO packet tracer, Bevywise IOT simulator, Ansys simulator, IBM Bluemix simulator etc. These simulators are used for showing simulation of the motes in Internet of Things (IoT). Among all these Cooja simulator is most popular simulator for IoT.

Operating System

There are three types of operating systems that support Internet of Things (IoT).

- Tiny OS(No support available as on date, no support for IoT devices)
- Contiki OS(So popular)
- RIOTOS(Upcoming and becoming popular).

Cooja Simulator is used for implementation in Contiki OS. Programming of Internet of Things is done in Contiki OS using Cooja Simulator. Cooja is the Contiki network simulator. Cooja authorizes large moreover little networks of Contiki motes to be replicated. Contiki is an effective organization that focuses on low power IoT devices.Motes can be emulated at the hardware level which is slower but allows precise inspection of the system behaviour, or at a less detailed level, which is faster and allows simulation of larger networks. Contiki is an open source operating system that is used to connect with the tiny low-cost microcontrollers and sensors to the internet. Contiki operating system is preferred because it supports various internet standards, rapid development, selection of hardware, active community to help and commercial support together with an open source license.

Contiki is designed mainly for tiny devices and thus the memory footprint is very less when compared with other systems, it supports the Full TCP with IPv6 and it handles power recognition where in the device power management is handled by the OS. All the modules of Contiki are loaded and unloaded during the run time. One of the most important features of Contiki is the use of Cooja Simulator to emulate if any of the hardware devices is not available.Contiki OS makes use of with Cooja simulator for the implementation of 6LoWPAN in IoT. Following figures show the screens of Contiki OS and simulation to send data from source to destination using Cooja simulator.

B.Manju Bashini
I B.Sc. (Computer Technology)

EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI)

Explainable artificial intelligence (XAI) is a set of processes and methods that allows human users to comprehend and trust the results and output created by machine learning algorithms. Explainable AI is used to describe an AI model, its expected impact and potential biases. It helps characterize model accuracy, fairness, transparency and outcomes in AI-powered decision making. Explainable AI is crucial for an organization in building trust and

confidence when putting AI models into production. AI explainability also helps an organization adopt a responsible approach to AI development.

As AI becomes more advanced, humans are challenged to comprehend and retrace how the algorithm came to a result. The whole calculation process is turned into what is commonly referred to as a “black box" that is impossible to interpret. These black box models are created directly from the data, and not even the engineers or data scientists who create the algorithm can understand or explain what exactly is happening inside them or how the AI algorithm arrived at a specific result.

There are many advantages to understanding how an AI-enabled system has led to a specific output. Explainability can help developers ensure that the system is working as expected, it might be necessary to meet regulatory standards, or it might be important in allowing those affected by a decision to challenge or change that outcome.

With explainable AI as well as interpretable machine learning organizations can gain access to AI technology’s underlying decision-making and are empowered to make adjustments. Explainable AI can improve the user experience of a product or service by helping the end user trust that the AI is making good decisions. As AI becomes more advanced, ML processes still need to be understood and controlled to ensure AI model results are accurate. Let’s look at the difference between AI and XAI, the methods and techniques used to turn AI to XAI and the difference between interpreting and explaining AI processes.

XAI implements specific techniques and methods to ensure that each decision made during the ML process can be traced and explained. AI, on the other hand, often arrives at a result using an ML algorithm, but the architects of the AI systems do not fully understand how the algorithm reached that result. This makes it hard to check for accuracy and leads to loss of control, accountability and auditability.

M.Harini

II B.Sc. (Computer Technology)

REINFORCEMENT LEARNING

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the

reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

Reinforcement Learning (RL) is the science of decision making. It is about learning the optimal behaviour in an environment to obtain maximum reward. In RL, the data is accumulated from machine learning systems that use a trial-and-error method. Data is not part of the input that we would find in supervised or unsupervised machine learning. Reinforcement learning uses algorithms that learn from outcomes and decide which action to take next. After each action, the algorithm receives feedback that helps it determine whether the choice it made was correct, neutral or incorrect. It is a good technique to use for automated systems that have to make a lot of small decisions without human guidance.

Reinforcement learning is an autonomous, self-teaching system that essentially learns by trial and error. It performs actions with the aim of maximizing rewards or in other words, it is learning by doing in order to achieve the best outcomes.

Types of Reinforcement

There are two types of Reinforcement:

Positive: Positive Reinforcement is defined as when an event, occurs due to a particular behaviour, increases the strength and the frequency of the behaviour. In other words, it has a positive effect on behaviour. Advantages of reinforcement learning are:

- Maximizes Performance
- Sustain Change for a long period of time
- Too much Reinforcement can lead to an overload of states which can diminish the results

Negative: Negative Reinforcement is defined as strengthening of behaviour because a negative condition is stopped or avoided. Advantages of reinforcement learning:

- Increases Behaviour
- Provide defiance to a minimum standard of performance

Elements of Reinforcement Learning

Reinforcement learning elements are as follows:

- Policy
- Reward function
- Value function
- Model of the environment

Policy: Policy defines the learning agent behaviour for given time period. It is a mapping from perceived states of the environment to actions to be taken when in those states.

Reward function: Reward function is used to define a goal in a reinforcement learning problem. A reward function is a function that provides a numerical score based on the state of the environment

Value function: Value functions specify what is good in the long run. The value of a state is the total amount of reward an agent can expect

to accumulate over the future, starting from that state.

Model of the environment: Models are used for planning.

- Credit assignment problem: Reinforcement learning algorithms learn to generate an internal value for the intermediate states as to how good they are in leading to the goal. The learning decision maker is called the agent. The agent interacts with the environment that includes everything outside the agent.

The agent has sensors to decide on its state in the environment and takes action that modifies its state.

- The reinforcement learning problem model is an agent continuously interacting with an environment. The agent and the environment interact in a sequence of time steps. At each time step t, the agent receives the state of the environment and a scalar numerical reward for the previous action, and then the agent then selects an action.

Reinforcement learning is a technique for solving Markov decision problems.

- Reinforcement learning uses a formal framework defining the interaction between a learning agent and its environment in terms of states, actions and rewards. This framework is intended to be a simple way of representing essential features of the artificial intelligence problem.

Various Practical Applications of Reinforcement Learning

- RL can be used in robotics for industrial automation.
- RL can be used in machine learning and data processing
- RL can be used to create training systems that provide custom instruction and materials according to the requirement of students.

Application of Reinforcement Learnings

1. Robotics: Robots with pre-programmed behaviour are useful in structured environments such as the assembly line of an automobile manufacturing plant, where the task is repetitive in nature.

2. A master chess player makes a move. The choice is informed both by planning, anticipating possible replies and counter replies.

3. An adaptive controller adjusts parameters of a petroleum refinery's operation in real time.

V.B Krishna Prabu

II B.Sc. (Computer Technology)

NEW PROGRAMMING LANGUAGES

Learning a new programming language does more than just educate users on one specific area of coding. It can also help them sharpen problem-solving skills, boost their job opportunities and get a better understanding of technology as a whole.

Functional Programming Languages

1. F#

F# is an open-source, cross-platform language that takes on more of a hybrid position between general and functional languages. Many programmers find F# to offer the same kind of simplicity as Python while delivering a more seamless experience than C# and Java. This may be because the language avoids the clutter of semicolons, curly brackets and other symbols, so developers don't have to worry about clarifying their object type. As a result, tasks such as list processing and applying complex type definitions are easier when working in F#. The hybrid nature of F# also makes it compatible with other styles, including databases, websites and .NET entities. Whatever elements designers are working with, they can rely on the programming language's strong type system to root out common errors. These factors all contribute to the flexibility and convenience of F#, which is why it remains a popular programming language.

2. Clojure

Clojure is a general-purpose language designed for concurrency, which means it supports multiple computations happening at the same time. But Clojure is also a Lisp language, keeping its syntax to a minimum. These elements facilitate a coding environment where developers can easily preserve code while building on previous projects to make changes as needed.

This programming language was also made for the Java Virtual Machine (JVM), so it pairs well with any system related to the JVM. It's no surprise then that many companies have added Closure to their tech stacks, including Adobe, Apple and Netflix.

3. Elixir

Elixir, however, is easier to write than Erlang, with the functional programming concepts of a language like Haskell. Elixir runs on the Erland virtual machine, which works well for low-latency distributed systems. The platform prioritizes scalability and fault tolerance. Lightweight threads of events, or processes, send messages to each other. Those processes can run concurrently, maximizing machine resources and making it easier to scale vertically or horizontally. If something goes wrong, the platform shows the developer the last known state that's sure to work.

4. PureScript

PureScript is a purely functional programming language that compiles to JavaScript. Most comparable to Haskell, PureScript is best used for developing web applications and server-side apps. Like Haskell, it uses algebraic data types, pattern matching and type classes. PureScript's types are expressive and support type inference, meaning that it requires far fewer explicit type annotations than other languages. One of its biggest strengths is its interoperability with other languages that target JavaScript.

5. Swift

Swift is a general-purpose compiled programming language developed by Apple that allows developers to write software for phones, servers, desktops or really anything else that runs on code. Originally developed as a replacement for Apple's earlier programming language, Objective-C, Swift combines ideas from other languages like Objective-C, Rust, Ruby and Python to help reduce common programming errors. The language combines a powerful type inference with a modernized syntax that helps ideas to be clearly expressed through code. Swift is an especially important skill for those seeking iOS developer roles.

Procedural Programming Languages

1. Go

Go is a C-style language created by engineering leads at Google. Sleeker than C++ or Java and more typesafe than Ruby or Python, Go comes with benefits and drawbacks. Some drawbacks: Typing is strict. One can't mix signed and unsigned integers, or integer sizes. Go also has some noticeable omissions: There are no generics and no inheritance. But Go's simplicity creates some marked advantages. Namely, the language is easy to use. There's less hiding behind the written code and the lack of inheritance helps developers avoid webs of dependencies, making it a solid language for data science. Tight definitions and thread safety seem to be Go priorities.

Object-Oriented Programming Languages

1. Dart

Another C-style language from Google, Dart is like JavaScript with type safety. It can easily compile to JavaScript, native machine code or WebAssembly. It can also run back-end code. Dart is good for building user interfaces with event-driven code. The hot reload command lets developers see changes to their applications instantaneously.

Some other Dart advantages, according to one Dart team member: optional static types, minimal compile-time errors and a strong, built-in editor.

2. Apache Groovy

Apache Groovy integrates with the Java platform and was made with the purpose of making life easier for Java developers. The programming language showcases concise and flexible syntax, allowing developers to reduce the time it takes to complete projects. This trait is also one of many reasons why Apache Groovy comes with a flat learning curve, rivaling the simplicity of languages like Python. Developers don't have to choose between static and dynamic languages since Apache Groovy supports both types. These features are what make Apache Groovy a great programming language for conducting tests. The syntax is designed to be test-friendly, leading many Java developers to embrace this language.

3. Crystal

Crystal is an object-oriented programming language that employs easy-to-learn syntax, especially for Ruby developers since the language takes its cue from Ruby's simple syntax. The language is also static, allowing it to catch errors earlier on in the development process. This feature spares teams from making expensive mistakes during runtime, such as overlooking null references. As an extra measure, Crystal provides built-in type inference, so developers don't have to clarify which language they're using every time. Crystal also supports concurrency with a fiber system, allowing developers to perform more computations without draining memory.

10. Pony

Pony is a language based on data-race-free typing and garbage collection, and uses the actor model as well as something called reference capabilities. Reference capabilities compel the programmer to label pieces of data as mutable, immutable or isolated. If data is mutable, the compiler doesn't allow the programmer to exchange the data between actors when two actors access mutable data at the same time, they may make contradictory updates, or the data could get corrupted. Reference capabilities keep data safe and eliminate the need for locks to prevent concurrent data updates. With no locks, concurrent programs run faster. Down-sides to Pony are low API stability, few high-quality third-party libraries and limited native tooling.

P.Logesh

II B.Sc. (Information Technology)

EXPLORING THE LATEST IN ROBOTICS TECHNOLOGY

The robotics industry is evolving at a rapid pace. Ground breaking advancements emerging in various sectors, from industrial automation to healthcare robotics. In this article, we give a curated recap of the latest trends and innovations in robotics that we have covered.

Humanoid Robots in the Factory

Humanoid robots are making strides towards integration in industry and healthcare. Recent developments by companies like Amazon and BMW involve testing humanoid robots within workplace settings. These robots aim to handle repetitive or unsafe tasks, enhance efficiency, and work alongside humans. Challenges include achieving safety and reliability levels.

New Mobile Robots on the Market

Autonomous mobile robots (AMRs) are revolutionizing various industries by offering flexible and intelligent solutions. Unlike automated guided vehicles (AGVs), which follow specific guided paths, AMRs use data from cameras, laser scanners and other sensors to navigate freely in work environments while avoiding obstacles. They can operate alongside human workers, enhancing efficiency.

3.A New Cobot Manufacturer

French company MS-Innov is making its entry into the manufacturing sector with its modular cobots. What sets MS-Innov's cobots apart are two key features:

Modularity: The MS-Innov cobots come in 4-axis, 5-axis and 6-axis configurations, available in two sizes (M and S) and three module types (M1, M2, and M3). This scalability and customization allow them to meet varying payload requirements and reach. The cobots can be assembled manually within minutes using an intuitive nut-and-bolt system. The focus is on flexibility, ensuring that once purchased, the cobot can be reconfigured for different application scenarios, avoiding idle equipment in companies.

Infinite Rotation: Unlike traditional systems, MS-Innov's cobots can rotate infinitely. They utilize pins to transmit both information and power, streamlining production processes by eliminating the need for repositioning.

Industrial Exoskeletons

Industrial exoskeletons are stepping off the silver screen and into the workplace. Unlike their cinematic counterparts, real-world exoskeletons prioritize worker safety over sheer strength. Around 93,000 exoskeletons were used in workplaces globally in 2022, with that number expected to increase sevenfold by 2030. These wearable devices assist workers in specific tasks, aiming to reduce injuries and fatigue.

M.S.K Manassha

II B.Sc. (Information Technology)

◆••••••••••••••••••••••••••••••••••◆

THE FUTURE OF CODING

Coding is advancing at a rapid pace, driven by new languages and ground breaking technologies. According to the **Bureau of Labor Statistics**, software development jobs are projected to grow from 2022 to 2032, much faster than the average for all occupations.

Understanding current coding trends is essential for several reasons:

- **Market demand:** Coding trends often reflect the current demands of the market. By keeping up with these trends, one can tailor one's skills and expertise to match what employers and clients are looking for. This increases one's employability and opportunities for freelance work.

- **Efficiency and effectiveness:** New coding trends often bring with them innovative tools, frameworks and methodologies that can significantly

improve development efficiency and effectiveness. By incorporating these trends into one's workflow, one can streamline processes, reduce development time and deliver higher-quality products.

- **Competitive advantage:** In today's competitive job market, staying ahead of coding trends can give one a competitive edge over other candidates. Employers value candidates who demonstrate awareness of the latest technologies and trends, as it shows a commitment to continuous learning and improvement.

- **Career growth:** Understanding coding trends allows one to anticipate future industry developments and position oneself for career growth. By acquiring expertise in emerging technologies, one can become a sought-after specialist in one's field and advance into higher-paying or more senior roles.

- **Innovation:** Coding trends often pave the way for innovation by introducing new concepts, ideas and approaches to software development. By staying informed about these trends, one can harness the latest innovations to create ground breaking solutions and push the boundaries of what's possible in one's field.

N.Lavanya

III B.Sc. (Information Technology)

THE ROLE OF ARTIFICIAL INTELLIGENCE (AI) IN THE METAVERSE

Artificial intelligence or AI and metaverse technology are the two most noticeable trends in discussions on technology in the 21st century. Each technology can potentially enhance the efficiency and productivity of different processes. Artificial intelligence and metaverse applications can offer new prospects for improving productivity in different industries such as gaming, education, healthcare and others. However, the technologies have been discussed separately without any idea about the relationship between them.

The metaverse's impact on society and AI's ability to manage complicated tasks required for the metaverse establish a synergetic relationship between them.

How Can Artificial Intelligence fit in the Metaverse?

Artificial intelligence, or AI, is a special domain of computer science that focuses on using natural language prompts as inputs for generating human-like actions. AI systems are capable of autonomous action, reasoning and learning according to programmed instructions. One can learn more about AI's role in the metaverse by identifying the core traits of artificial intelligence which align with

the metaverse. AI research focuses on developing machines that can process information and understand natural language.

Machines would process data and take decisions like human beings and they achieve such functionalities by processing the data generated by people every day. Another significant highlight of AI points to its ability for faster and more efficient data processing. Artificial intelligence applications in the metaverse would rely on the capabilities of machine learning to use data generated in the metaverse. AI could process data to identify patterns and learn from the patterns to improve their performance.

Interestingly, people use simple AI systems in their everyday lives to obtain information. For example, recommendations based on product searches have machine learning working their wonder behind the scenes. As of now, most of the research on artificial intelligence has been focused on improving the relevance of AI for users. The common theme in research on artificial intelligence revolves around understanding human behaviour and the physical world. The prospects for an AI-generated metaverse would translate into reality in the future. It is important to note the changing trends in computing, focused on contextual rather than static experiences. The devices around us are gradually adapting and becoming better at understanding and anticipating our needs because of AI.

How will the Metaverse Welcome AI?

The fundamentals of artificial intelligence provided a brief overview of how the core traits of AI could support metaverse technology. Now, one can learn more about the metaverse and Artificial Intelligence equation by exploring the basics of the metaverse. The Metaverse technology represents an open, shared and persistent three-dimensional world, which bridges the gap between virtual and physical worlds. Some people have also described the metaverse as a 3D version of the Internet. The metaverse involves a combination of different technologies, including physical, virtual and augmented reality, in shared online space.

The advantages of metaverse AI use cases could help in capitalizing on the data generated from digital activity in virtual spaces. Metaverse technology provides immersive experiences with the help of intuitive interfaces, which allow users to create and interact with visual information. AI could help in improving the speed of development of the metaverse. Although many companies are actively involved in creating their own metaverse platforms, the actual vision for the metaverse suggests otherwise. The metaverse should be one universal platform accessible to every individual without any walled gardens.

S.Dinesh

III B.Sc. (Computer Technology)

CODING TRENDS 2024

Artificial Intelligence

Artificial Intelligence (AI) is truly transforming the coding world. Python, with its powerful libraries like TensorFlow and PyTorch, makes building AI and machine learning applications much easier. But AI isn't just about creating smart apps; it's also one's new best friend in coding. Additionally, OpenAI's ChatGPT, for example, is revolutionizing how developers interact with code. This intelligent AI chatbot can assist in problem-solving, offering suggestions and even completing lines of code, making coding faster and more efficient than ever before. This means less time spent on repetitive tasks and more on the creative problem-solving that makes coding fun. AI is making coding faster, smarter and more efficient.

No-code

Just as AI is enhancing coding efficiency, no-code platforms are making development accessible to everyone. One noteworthy player in the no-code arena is Fuzen, a specialized platform tailored for building SaaS (Software as a Service) applications. With Fuzen, users can effortlessly create custom SaaS solutions without the need for coding expertise. This empowers entrepreneurs and businesses to bring their ideas to life quickly and cost-effectively. This trend is a game-changer because it lowers the barrier to entry for software development. Businesses can quickly prototype and launch solutions, staying ahead of market demands. For developers, it means more time to focus on complex tasks while empowering others to turn their ideas into reality. No-Code is all about speed, innovation and accessibility.

Blockchain applications

Blockchain is revolutionizing security and transparency. Blockchain technology is about more than just cryptocurrencies; it's a game-changer for creating secure, decentralized applications (dApps). If one has played around with Ethereum, one knows about Solidity, the language for writing smart contracts. And now, languages like Rust and Move are stepping up to enhance blockchain performance and security. Blockchain ensures data integrity without needing intermediaries, which is a big deal for industries like finance, healthcare, and supply chain management. For developers, it opens up a world of opportunities to build secure and reliable applications, making it an exciting area to explore.

Sustainable software development

There's also a growing focus on Sustainable Software Development in 2024. The tech industry's carbon footprint is a significant concern, with data centers consuming a lot of energy. Developers are now focusing on writing energy-efficient code and optimizing algorithms to reduce resource

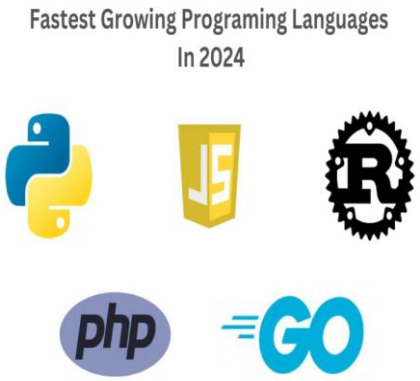
usage. Using energy-efficient languages and adopting green computing techniques helps the environment and improves software performance and cuts costs. As awareness of sustainability grows, expect more tools and frameworks designed to support eco-friendly coding practices. It's a win-win for the planet and our software.

Serverless computing (FaaS)

To focus on sustainability, Serverless Computing, or Function as a Service (FaaS), is transforming how we deploy applications. Platforms like AWS Lambda, Google Cloud Functions and Azure Functions let one run code without worrying about managing servers. Serverless computing automatically scales and saves costs since one only pays for the compute time one uses. This simplifies development, allowing one to focus on writing code and creating features rather than managing infrastructure. Serverless architectures lead to faster development cycles and more responsive applications.

Programming and programming languages

JavaScript (JS) remains a powerhouse in web development, powering dynamic and interactive websites. Its versatility and widespread adoption make it a must-know language for any developer looking to build modern web applications.\



Python continues to soar in popularity due to its simplicity, readability and vast ecosystem of libraries and frameworks. From web development to data science and machine learning, Python's flexibility makes it a favorite among developers of all skill levels.

Ruby is beloved for its elegant syntax and developer-friendly features. While its usage may have declined slightly in recent years, Ruby still maintains a dedicated community and remains a top choice for building web applications, particularly with the Ruby on Rails framework.

Go (Golang), a relatively new language developed by Google, has been gaining traction for its simplicity, performance and built-in support for concurrency. With its focus on simplicity and efficiency, Go is becoming increasingly popular for building scalable and high-performance backend systems.

Rust is another rising star in the programming world, known for its focus on safety,

performance and concurrency. Rust's memory safety features make it ideal for systems programming, where reliability and security are paramount.

Each of these languages brings its own unique strengths to the table, catering to different use cases and developer preferences. Whether one is building web applications, system software, or machine learning models, there's a programming language out there to suit one's needs.

Cybersecurity

Among all these advancements, cybersecurity remains a top priority. With cyber threats becoming increasingly sophisticated, developers need to ensure that their applications are secure from the ground up. Continuous monitoring, secure coding practices, and robust backup solutions are essential to keep data safe. Developers must integrate security measures throughout the entire development cycle and stay updated on the latest cybersecurity trends and threats. By addressing cybersecurity concerns proactively, developers can mitigate risks and safeguard sensitive data from potential breaches and attacks.

S.Dinesh

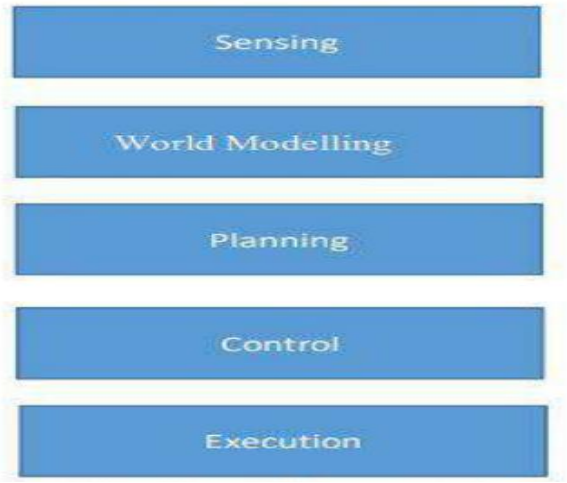
III B.Sc. (Computer Technology)

SPACE ROBOTICS

Space robotics is the development of general purpose machines that are capable of surviving in the space environment, performing exploration, construction, maintenance, servicing or other tasks. Humans control space robots from either a "local" control console or "remotely" controlled from human operators on Earth. Space robots are generally designed to do multiple tasks.

Space Research: "SPACE", the word itself signifies something infinite. Space travel has always been dangerous and any unexpected event can cause death. It is here that the robots play a huge role and help mankind in his research process. How Robots Work in Space? Working principle of Space robots are based on the SPA algorithm. SPA stands for sense, plan and action. It is used in built world modules to match and worked accordingly.

Flowchart



Technologies Used

Mapping and navigation One of the basic functions of a space robot is to navigate its way cleverly through all obstacles that come in its way. Mapping and navigation comprise of three more technologies.

1.Obstacle avoidance

2.Mapping

3.Path planning

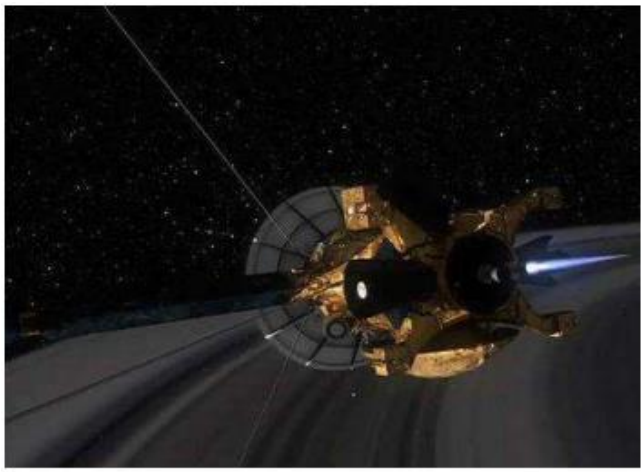
4.Planning: It is a feature by which a robot understands the situation and

5.decides a strategy to tackle it.

6.Sequencing: Selection of a particular skill set which would result in perfect execution of a plan. 7.Control: Performing the selected skill set to perfection.

Types of Space Robots

1. **Planetary Rovers:** It is the most advanced form of robotics technology used in space research. They are the robots, which explore, navigate and research themselves with the least human intervention; they analyse the data collected and send the results back to earth.



2.IN-Orbit Operators: They are the robots, which assist an astronaut during his space mission. For example, a robot can be designed specially to refuel a shuttle thus helping the astronaut to remain in his shuttle and accomplish various tasks without any risk to their lives.

3.Probes: A similar class of robots explores the system without actually physically landing anywhere. These typically use cameras and variety of instruments to measure other planets, moons, and the sun from distance. Most of these use solar cells to their instruments.

Patient is the key element of success.

-Bill Gates