**FEDERATED LEARNING-BASED 3D MEDICAL IMAGE COMPRESSION**

**Abstract:**

This project presents a **Federated Learning-Based 3D Medical Image Compression** system that combines **Optimal Multi-linear Singular Value Decomposition (OMLSVD)** and **deep auto-encoders** to compress 3D medical images with minimal quality loss. Traditional compression methods like JPEG2000 reduce image quality after decompression, while our method ensures high fidelity and reduced storage. To preserve data privacy, **Federated Learning** is used—each medical client trains a local model and shares only the weights with a global server, avoiding direct data sharing. The system is tested on a 3D chest X-ray dataset and shows improved performance in terms of **SSIM**, **PSNR**, and storage size. Results confirm that the proposed method offers better compression, high-quality decompression, and privacy-preserving collaboration across clients.

**Introduction:**

Medical imaging technologies such as Magnetic Resonance Imaging (MRI), Computed Tomography (CT), and Positron Emission Tomography (PET) have become essential tools in modern clinical diagnosis and treatment planning. These technologies generate high-resolution, three-dimensional (3D) images that offer detailed anatomical insights, but they also produce extremely large volumes of data. Efficient storage, transmission, and processing of this data are critical challenges for hospitals and research institutions, especially in resource-constrained settings. Traditional image compression methods often involve lossy techniques that can compromise diagnostic quality, or require centralized data aggregation, which raises serious privacy and security concerns.

In recent years, the integration of deep learning techniques, particularly convolutional neural networks (CNNs) and autoencoders, has significantly advanced the state-of-the-art in medical image compression. However, training these models typically demands access to large, diverse, and high-quality datasets. In the healthcare domain, such centralized data collection is often infeasible due to strict regulations like HIPAA (Health Insurance Portability and Accountability Act) and GDPR (General Data Protection Regulation), which prohibit the free sharing of patient data. This limitation creates a pressing need for a privacy-preserving approach that enables collaborative model training without compromising data confidentiality.

Federated Learning (FL) has emerged as a powerful solution to this problem by allowing multiple decentralized clients—such as hospitals or imaging centers—to collaboratively train a global model without transferring raw data. In a federated learning setup, each client trains a local model on its own data and only shares encrypted model updates or gradients with a central server, which aggregates them to improve the shared model. This approach ensures that patient data remains local and secure while still benefiting from the collective intelligence of distributed datasets. Applying FL to the domain of 3D medical image compression offers the potential to build highly effective, generalizable compression models that maintain diagnostic fidelity, support interoperability, and comply with privacy regulations, thereby advancing both medical research and clinical applications.

**Literature Survey:**

### **1. Title: Federated Learning: Strategies for Improving Communication Efficiency**

**Author(s):** Kairouz et al. (2021)  
**Description:**  
This seminal work provides a comprehensive overview of federated learning frameworks, challenges, and communication-efficient strategies. The authors discuss various aggregation techniques, such as Federated Averaging (FedAvg), and explore methods to reduce communication costs while maintaining model performance. This research forms the foundation for implementing FL in bandwidth-limited healthcare environments where large-scale 3D medical imaging data must be processed efficiently and securely.

### **2. Title: Deep Learning-Based Image Compression for Medical Applications**

**Author(s):** Chen et al. (2020)  
**Description:**  
The authors present an autoencoder-based approach for compressing 2D and 3D medical images. The model achieves a high compression ratio while preserving the diagnostic integrity of the images. This study is critical in demonstrating the potential of neural network architectures, such as convolutional autoencoders, in compressing complex volumetric data without the need for handcrafted features or traditional codecs.

### **3. Title: Privacy-Preserving Deep Learning for Medical Image Analysis Using Federated Learning**

**Author(s):** Sheller et al. (2020)  
**Description:**  
This study explores federated learning as a secure alternative to centralized deep learning for medical image analysis. Using datasets from multiple institutions, the authors demonstrate that FL can achieve comparable accuracy to centralized models while preserving data privacy. This work serves as direct inspiration for federated learning applications in medical image compression, particularly in multi-institutional settings.

### **4. Title: A Survey on Compression Techniques for Medical Images**

**Author(s):** Singh et al. (2019)  
**Description:**  
The survey outlines traditional and modern techniques for medical image compression, including lossless and lossy methods. It highlights the trade-offs between compression efficiency and image quality, especially in sensitive applications like diagnostics. The paper provides context on why deep learning and federated approaches are becoming more relevant in the field**.**

### **5. Title: Federated Learning for Healthcare Informatics**

**Author(s):** Li et al. (2021)  
**Description:**  
This paper reviews federated learning's role in healthcare, discussing its applications in diagnosis, patient monitoring, and medical imaging. It emphasizes the challenges of heterogeneity in client data and model personalization. The insights from this paper support the adoption of federated learning in scenarios where 3D imaging data varies across institutions in terms of resolution, modality, and patient demographics.

### **6. Title: 3D Medical Image Segmentation Using Deep Neural Networks: A Review**

**Author(s):** Litjens et al. (2017)  
**Description:**  
Although focused on segmentation, this review is valuable for understanding how deep neural networks handle 3D medical imaging data. It explains the architecture and training considerations for 3D CNNs, which are also applicable to compression tasks. The paper supports the design of deep learning models tailored to volumetric data in a federated setting.

**Existing System:**

Traditional 3D medical image compression techniques such as PCA, Discrete Wavelet Transform (DWT), and JPEG2000 are commonly used to reduce storage requirements. However, these methods often lead to significant loss in image quality upon decompression, which is critical in medical diagnostics. Furthermore, when deep learning methods like auto-encoders are used for compression, they require centralized data training, raising privacy concerns, especially when sensitive patient data from multiple medical institutions are involved.

**Disadvantages of Existing Systems:**

### 1. **Privacy and Security Concerns**

Traditional systems rely on centralized data collection, which requires hospitals and medical institutions to upload sensitive patient data to a central server. This process increases the risk of data breaches, unauthorized access, and non-compliance with data protection regulations such as HIPAA and GDPR. Even anonymized data can be susceptible to re-identification attacks when combined with external sources.

### 2. **High Communication and Storage Overhead**

Transmitting large volumes of 3D medical imaging data (such as MRI and CT scans) over networks to a central location demands high bandwidth and storage infrastructure. This not only slows down the training process but also makes the system less scalable, particularly in resource-constrained settings or rural healthcare centers.

### 3. **Limited Scalability and Generalization**

Compression models trained on centralized datasets often suffer from limited generalizability. If the data is biased toward specific demographics, imaging equipment, or clinical protocols, the model may perform poorly on unseen data from diverse sources. Additionally, expanding the system to include new participants requires additional data integration and retraining, which is computationally expensive.

### 4. **Single Point of Failure**

Centralized systems are vulnerable to single points of failure. If the central server experiences downtime, is attacked, or becomes corrupted, the entire system's functionality can be compromised. This reduces the robustness and reliability of the solution in real-world medical environments.

### 5. **Regulatory and Ethical Challenges**

Many countries have strict legal and ethical frameworks that limit the sharing of patient data across institutions or borders. Centralized compression systems often struggle to meet these requirements, making their deployment legally complicated or altogether infeasible in certain regions.

**Proposed System:**

The proposed system introduces a hybrid compression approach using **Optimal Multi-linear Singular Value Decomposition (OMLSVD)** along with **deep auto-encoders** to achieve high-quality 3D medical image compression with lower storage usage. To address privacy concerns, **Federated Learning** is integrated, allowing multiple medical clients to train models locally and share only the model weights with a centralized federated server. This ensures data privacy while enabling collaborative learning. The system significantly improves decompressed image quality (in terms of SSIM and PSNR) and reduces storage, making it suitable for secure, scalable medical image management.

**Advantages of the Proposed System:**

### 1. **Enhanced Data Privacy and Security**

By keeping all raw 3D medical images on local devices and only sharing encrypted model updates, the proposed federated learning framework ensures that sensitive patient data is never exposed or transferred. This significantly reduces the risk of data breaches and fully aligns with privacy regulations like HIPAA and GDPR.

### 2. **Compliance with Legal and Ethical Standards**

The system supports decentralized learning, which is ideal for healthcare environments with strict legal constraints on data sharing. It enables collaborative model training across multiple institutions without violating ethical guidelines or regulatory frameworks.

### 3. **Scalability Across Institutions**

New healthcare centers and diagnostic labs can be seamlessly integrated into the federated learning network without requiring access to existing data or retraining from scratch. This makes the system highly scalable and adaptable to dynamic, multi-institutional environments.

### 4. **Reduced Communication and Storage Overhead**

Instead of transferring large 3D imaging datasets, only compact model parameters or gradients are transmitted, significantly minimizing bandwidth usage and storage demands. This makes the system suitable for use in bandwidth-constrained or rural healthcare settings.

### 5. **High Compression Efficiency with Diagnostic Quality**

The use of deep learning-based 3D autoencoders allows for high compression ratios while preserving critical anatomical details and diagnostic accuracy. Evaluation metrics such as PSNR and SSIM demonstrate the system’s ability to maintain high image fidelity post-compression.

### 6. **Improved Model Generalization**

Since the global model is trained on updates from diverse datasets across various institutions, it becomes more robust and generalizes better across different imaging modalities, patient demographics, and scanner types, reducing bias and increasing clinical reliability.

**System Analysis:**

The healthcare sector generates vast amounts of 3D medical imaging data from modalities like MRI, CT, and PET scans. Efficient compression of these images is critical to reduce storage requirements and enable faster transmission between medical facilities. Traditional centralized compression systems require uploading large volumes of sensitive patient data to a central server for training deep learning models. This centralized approach raises serious privacy concerns, as it exposes data to potential breaches and may violate strict healthcare regulations such as HIPAA and GDPR. Furthermore, centralized systems face challenges like high communication costs, limited scalability, and a single point of failure that can disrupt the entire system if the central server encounters issues.

To overcome these challenges, the proposed system leverages federated learning, which enables collaborative training of compression models without sharing raw data. Each medical institution retains its own dataset and trains the compression model locally. Only encrypted model updates are shared with a central server, which aggregates these updates to improve the global model. This decentralization directly addresses privacy concerns and complies with regulatory requirements, as sensitive medical images never leave their source locations. Additionally, this approach reduces the risk of data misuse or breaches, making it a safer and more ethical choice for handling patient data.

The system’s architecture involves multiple clients (medical institutions), each equipped with local storage and computing resources for training 3D autoencoder-based compression models on their own datasets. After each training cycle, clients send model parameters to a central aggregator that uses algorithms like Federated Averaging to combine updates and produce a global model. This model is redistributed to clients for further training iterations. To enhance security, the system employs advanced techniques such as differential privacy and secure aggregation, ensuring that shared model updates cannot be reverse-engineered to reveal private data. This layered approach enhances trust and reliability among participating institutions.

Performance evaluation of the system considers not only traditional image compression metrics—such as compression ratio, Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM)—but also federated learning-specific parameters like communication efficiency, model convergence speed, and the impact of data heterogeneity across clients. Balancing these factors is crucial to maintain high-quality image compression while ensuring practical deployment in diverse clinical settings. The system is optimized to provide consistent and accurate image reconstructions that retain diagnostically relevant details, all while minimizing the overhead related to data exchange and computational load.

Despite its advantages, the proposed system faces some limitations. Data variability between institutions can slow model convergence and affect the uniformity of compression performance. While federated learning reduces data transmission, communication of model updates still demands bandwidth and can introduce delays, especially in networks with limited connectivity. Future improvements may include personalized federated models tailored to individual client data characteristics, adaptive synchronization intervals to reduce communication load, and integration with edge computing for local inference. Continuous experimentation with real-world clinical data will help refine these approaches and improve the robustness and usability of federated learning-based 3D medical image compression systems.

**SYSTEM REQUIREMENTS:**

HARDWARE REQUIREMENTS:

• System : Pentium IV 2.4 GHz.

• Hard Disk : 40 GB.

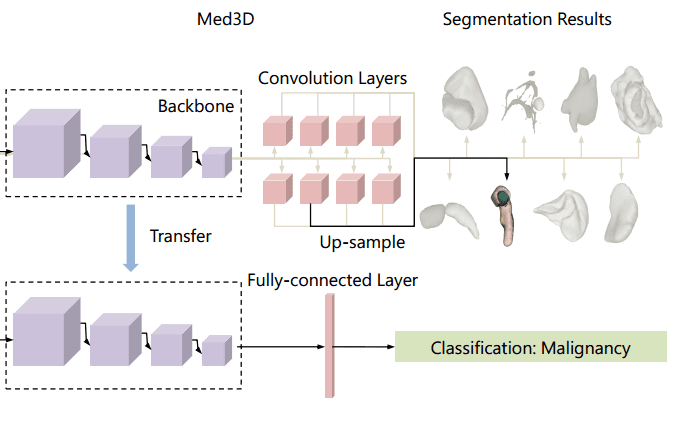
• Ram : 512 Mb.

SOFTWARE REQUIREMENTS:

• Operating system : - Windows.

• Coding Language : python.

**System Architecture**



**UML Diagrams:**

**UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:** The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**Class diagram**

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram was capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

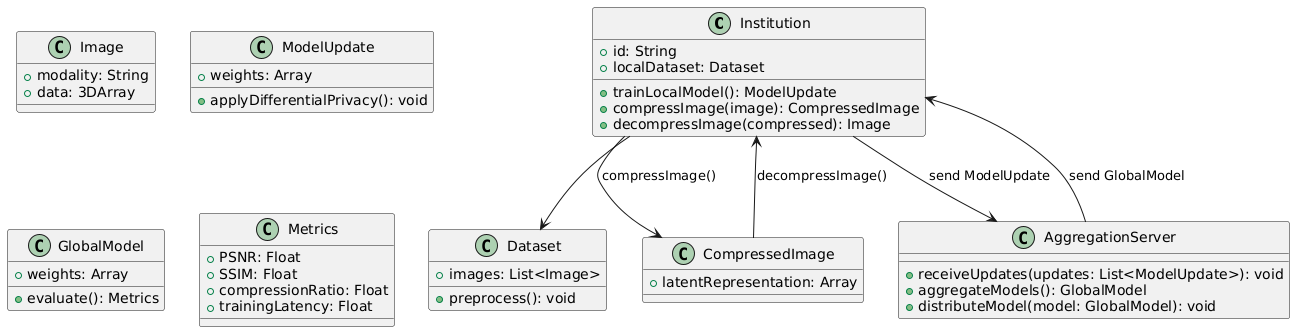


Figure-5.1: Class Diagram

**Sequence Diagram**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines (“lifelines”), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

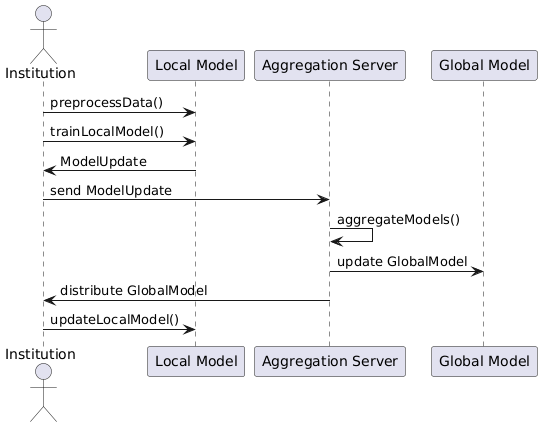


Figure-5.2: Sequence Diagram

**Activity diagram**

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

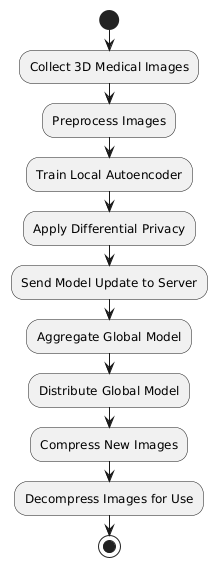


Figure-5.3: Activity Diagram

**Data flow diagram**

A data flow diagram (DFD) is a graphical representation of how data moves within an information system. It is a modeling technique used in system analysis and design to illustrate the flow of data between various processes, data stores, data sources, and data destinations within a system or between systems. Data flow diagrams are often used to depict the structure and behavior of a system, emphasizing the flow of data and the transformations it undergoes as it moves through the system.

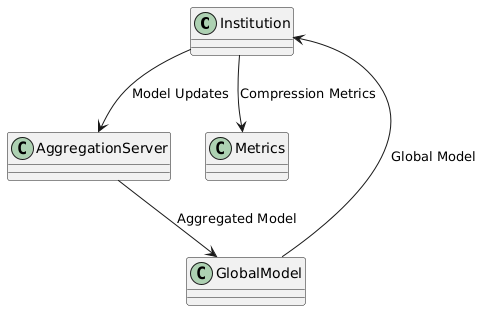


Figure-5.4: Dataflow Diagram

**Component diagram:** Component diagram describes the organization and wiring of the physical components in a system.

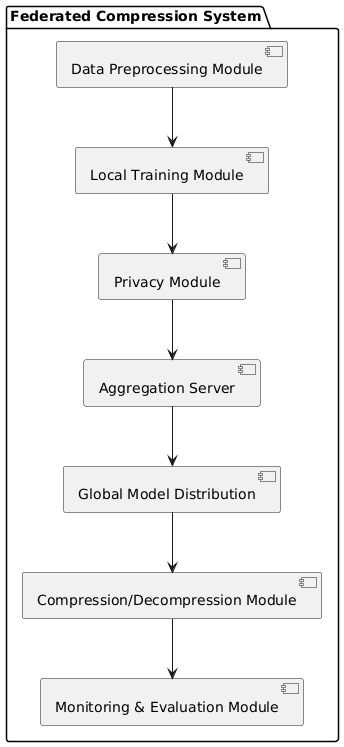


Figure-5.5: Component Diagram

**Use Case diagram:** A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

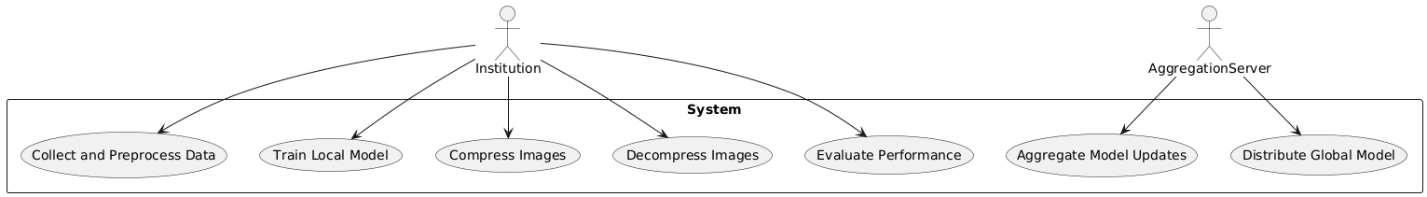


Figure-5.6: Use Case Diagram

**Deployment Diagram:**

A deployment diagram in UML illustrates the physical arrangement of hardware and software components in the system. It visualizes how different software artifacts, such as data processing scripts and model training components, are deployed across hardware nodes and interact with each other, providing insight into the system’s infrastructure and deployment strategy.

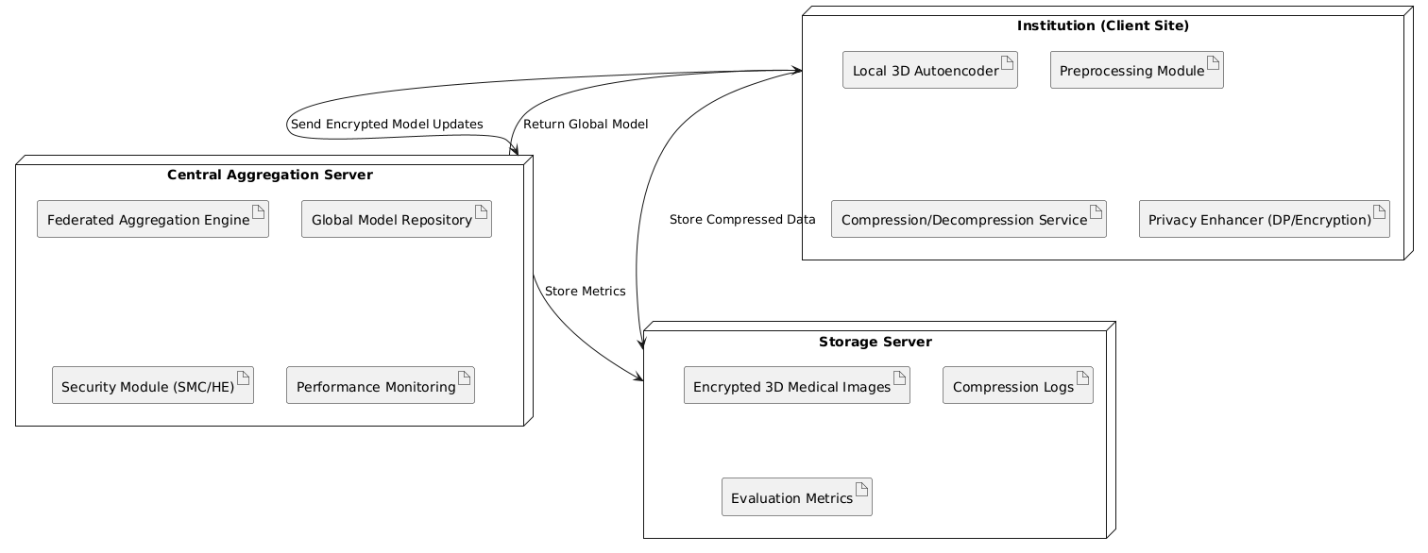


Figure-5.7: DeploymentDiagram

**System Implementations:**

### 1. **Data Collection and Preprocessing**

The system begins with collecting 3D medical imaging data such as MRI, CT, or PET scans from multiple healthcare institutions. Each institution stores its imaging datasets locally to ensure data privacy. Before training, the images undergo preprocessing steps including normalization, resizing, and augmentation to standardize input dimensions and improve model robustness. These steps help reduce noise and prepare the data for efficient compression.

### 2. **Local Model Training**

Each participating client (hospital or medical center) hosts a local instance of a 3D convolutional autoencoder designed for medical image compression. Using its own preprocessed datasets, the client trains the model locally to learn compressed representations of the images. The model compresses the input into a lower-dimensional latent space and then reconstructs the image during decoding. Local training ensures that patient data never leaves the institution, maintaining strict privacy compliance.

### 3. **Federated Model Aggregation**

After completing local training epochs, each client securely sends its model updates—such as weight parameters or gradients—to a central aggregation server. The server combines these updates using algorithms like Federated Averaging (FedAvg) to update the global compression model. This aggregated model benefits from the knowledge gained from multiple institutions without requiring access to any raw image data. The updated global model is then redistributed back to all clients for further training rounds.

### 4. **Security and Privacy Enhancements**

To prevent potential information leakage through shared model updates, the system integrates privacy-preserving techniques such as differential privacy, which adds noise to model updates, and secure multiparty computation or homomorphic encryption to enable encrypted aggregation. These mechanisms ensure that individual client data remains confidential and protected even during the collaborative learning process.

### 5. **Compression and Decompression Workflow**

Once the global model reaches satisfactory performance, it is deployed across client sites to perform real-time compression and decompression of 3D medical images. The model compresses large image volumes into compact latent representations, reducing storage space and enabling faster data transmission. Decompression reconstructs images with minimal loss in quality, maintaining essential diagnostic details. This workflow allows healthcare providers to efficiently manage imaging data while adhering to privacy regulations.

### 6. **Performance Monitoring and Evaluation**

The system includes tools to monitor compression quality, model convergence, and communication efficiency throughout the federated training process. Metrics like PSNR, SSIM, compression ratio, and training latency are logged and analyzed to ensure that the model maintains high diagnostic accuracy. Regular evaluation helps in fine-tuning the model and detecting any performance degradation or anomalies across participating clients.

**System Environment:**

# What is Python :-

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following .

* + [Machine Learning](https://www.geeksforgeeks.org/machine-learning/)
  + GUI Applications (like Kivy, Tkinter, PyQt etc. )
  + Web frameworks like Django (used by YouTube, Instagram, Dropbox)
  + Image processing (like Opencv, Pillow)
  + Web scraping (like Scrapy, BeautifulSoup, Selenium)
  + Test frameworks
  + Multimedia

### **Advantages of Python :-**

Let’s see how Python dominates over other languages.

#### 1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

#### 2. Extensible

As we have seen earlier, Python can be**extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

#### 3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities**to our code in the other language.

#### 4. Improved Productivity

The language’s simplicity and extensive libraries render programmers**more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

#### 5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

#### 6. Simple and Easy

When working with Java, you may have to create a class to print **‘Hello World’**. But in Python, just a print statement will do. It is also quite **easy to learn, understand,** and**code.** This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

#### 7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory.** This further aids the readability of the code.

#### 8. Object-Oriented

This language supports both the **procedural and object-oriented**programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

#### 9. Free and Open-Source

Like we said earlier, Python is **freely available.** But not only can you[**download Python**](https://data-flair.training/blogs/install-python-windows/) for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

#### 10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to**code only once**, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

#### 11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

### **Advantages of Python Over Other Languages**

#### 1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

#### 2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

**The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.**

#### 3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and [**machine learning**](https://data-flair.training/blogs/machine-learning-tutorials-home/), automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

### **Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

#### 1. Speed Limitations

We have seen that Python code is executed line by line. But since [Python](https://www.python.org/) is interpreted, it often results in **slow execution**. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

#### 2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the **client-side**. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

#### 3. Design Restrictions

As you know, Python is **dynamically-typed**. This means that you don’t need to declare the type of variable while writing the code. It uses **duck-typing**. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can**raise run-time errors**.

#### 4. Underdeveloped Database Access Layers

Compared to more widely used technologies like **JDBC (Java DataBase Connectivity)** and **ODBC (Open DataBase Connectivity)**, Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

#### 5. Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**History of Python : -**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python.Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI).

I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it."Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

**What is Machine Learning : -**

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data.

Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

**Categories Of Machine Leaning :-**

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction.

Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

## Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

## Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

**Quality of data** − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

**Time-Consuming task** − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

**Lack of specialist persons** − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

**No clear objective for formulating business problems** − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

**Issue of overfitting & underfitting** − If the model is overfitting or underfitting, it cannot be represented well for the problem.

**Curse of dimensionality** − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

**Difficulty in deployment** − Complexity of the ML model makes it quite difficult to be deployed in real life.

## Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

* Emotion analysis
* Sentiment analysis
* Error detection and prevention
* Weather forecasting and prediction
* Stock market analysis and forecasting
* Speech synthesis
* Speech recognition
* Customer segmentation
* Object recognition
* Fraud detection
* Fraud prevention
* Recommendation of products to customer in online shopping

# How to Start Learning Machine Learning?

Arthur Samuel coined the term **“Machine Learning”** in 1959 and defined it as a **“Field of study that gives computers the capability to learn without being explicitly programmed”.**

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to [Indeed](http://blog.indeed.com/2019/03/14/best-jobs-2019/), Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of **$146,085** per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

### **How to start learning ML?**

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

### Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

#### (a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

#### (b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!  
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

#### (c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is [Python](https://www.geeksforgeeks.org/python-programming-language/)! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as [Keras](https://keras.io/" \t "_blank), [TensorFlow](https://www.tensorflow.org/), [Scikit-learn](https://scikit-learn.org/stable/), etc.

So if you want to learn ML, it’s best if you learn Python! You can do that using various online resources and courses such as [**Fork Python**](https://practice.geeksforgeeks.org/courses/fork-python) available Free on GeeksforGeeks.

### **Step 2 – Learn Various ML Concepts**

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It’s best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

#### (a) Terminologies of Machine Learning

* **Model –**A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
* **Feature –**A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
* **Target (Label) –**A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
* **Training –**The idea is to give a set of inputs(features) and it’s expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
* **Prediction –**Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

#### (b) Types of Machine Learning

* **Supervised Learning –**This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
* **Unsupervised Learning –**This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
* **Semi-supervised Learning –**This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
* **Reinforcement Learning –**This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

### **Advantages of Machine learning :-**

#### 1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

#### 2. No human intervention needed (automation)

With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

#### 3. Continuous Improvement

As [**ML algorithms**](https://data-flair.training/blogs/machine-learning-algorithms/) gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

#### 4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

#### 5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

### **Disadvantages of Machine Learning :-**

#### 1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

#### 2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

#### 3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

#### 4. High error-susceptibility

[Machine Learning](https://en.wikipedia.org/wiki/Machine_learning) is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

**Python Development Steps : -**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.  
Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked.Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode.Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x.

The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

* Print is now a function
* Views and iterators instead of lists
* The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
* There is only one integer type left, i.e. int. long is int as well.
* The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
* Text Vs. Data Instead Of Unicode Vs. 8-bit

**Purpose :-**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Modules Used in Project :-**

**Tensorflow**

TensorFlow is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library for dataflow and differentiable programming](https://en.wikipedia.org/wiki/Library_(computing)) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache 2.0](https://en.wikipedia.org/wiki/Apache_License) [open-source license](https://en.wikipedia.org/wiki/Open-source_license) on November 9, 2015.

**Numpy**

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](http://ipython.org/) shells, the [Jupyter](http://jupyter.org/) Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the [sample plots](https://matplotlib.org/tutorials/introductory/sample_plots.html) and [thumbnail gallery](https://matplotlib.org/gallery/index.html).

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels.

All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Install Python Step-by-Step in Windows and Mac :**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

## How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

**Note:** The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. [Download the Python Cheatsheet here.](https://myelearninghub.com/python-cheat-sheet/)The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

### Download the Correct version into the system

**Step 1:** Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: [https://www.python.org](https://www.python.org/)



Now, check for the latest and the correct version for your operating system.

**Step 2:** Click on the Download Tab.

****

**Step 3:** You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

****

**Step 4:** Scroll down the page until you find the Files option.

**Step 5:** Here you see a different version of python along with the operating system.



• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

**Note:** To know the changes or updates that are made in the version you can click on the Release Note Option.

### Installation of Python

**Step 1:** Go to Download and Open the downloaded python version to carry out the installation process.



**Step 2:** Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



**Step 3:** Click on Install NOW After the installation is successful. Click on Close.



With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

**Note:** The installation process might take a couple of minutes.

### Verify the Python Installation

**Step 1:** Click on Start

**Step 2:** In the Windows Run Command, type “cmd”.



**Step 3:** Open the Command prompt option.

**Step 4:** Let us test whether the python is correctly installed. Type **python –V** and press Enter.



**Step 5:** You will get the answer as 3.7.4

**Note:** If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

### Check how the Python IDLE works

**Step 1:** Click on Start

**Step 2:** In the Windows Run command, type “python idle”.



**Step 3:** Click on IDLE (Python 3.7 64-bit) and launch the program

**Step 4:** To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



**Step 5:** Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

**Step 6:** Now for e.g. **enter print**

**SYSTEM TEST**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### **TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Test cases1:**

**Test case for Login form:**

|  |  |
| --- | --- |
| **FUNCTION:** | **LOGIN** |
| **EXPECTED RESULTS:** | Should Validate the user and check his existence in database |
| **ACTUAL RESULTS:** | Validate the user and checking the user against the database |
| **LOW PRIORITY** | **No** |
| **HIGH PRIORITY** | **Yes** |

**Test case2:**

**Test case for User Registration form:**

|  |  |
| --- | --- |
| **FUNCTION:** | **USER REGISTRATION** |
| **EXPECTED RESULTS:** | Should check if all the fields are filled by the user and saving the user to database. |
| **ACTUAL RESULTS:** | Checking whether all the fields are field by user or not through validations and saving user. |
| **LOW PRIORITY** | **No** |
| **HIGH PRIORITY** | **Yes** |

**Test case3:**

**Test case for Change Password:**

When the old password does not match with the new password ,then this results in displaying an error message as “ OLD PASSWORD DOES NOT MATCH WITH THE NEW PASSWORD”.

|  |  |
| --- | --- |
| **FUNCTION:** | **Change Password** |
| **EXPECTED RESULTS:** | Should check if old password and new password fields are filled by the user and saving the user to database. |
| **ACTUAL RESULTS:** | Checking whether all the fields are field by user or not through validations and saving user. |
| **LOW PRIORITY** | **No** |
| **HIGH PRIORITY** | **Yes** |

**SCREEN SHOTS**

Federated Learning-Based 3D Medical Image Compression

In propose work we are employing Optimal Multi-linear Singular Value Decomposition (OMLSVD) and deep auto-encoders to compress medical 3D images. 3D medical images contains accurate information about diseases but this require heavy storage and to reduce storage size many compression algorithms were introduced such as PCA, DWT, JPEGCompreesion2000 and many more. All existing algorithms decompress image quality is very low.

To enhance 3D compression image quality we are combining OMLSVD and auto-encoder where OMLSVD will be utilize to compress image and then compress image along with normal image will get trained with auto-encoder algorithm to decompress compress image with high quality.

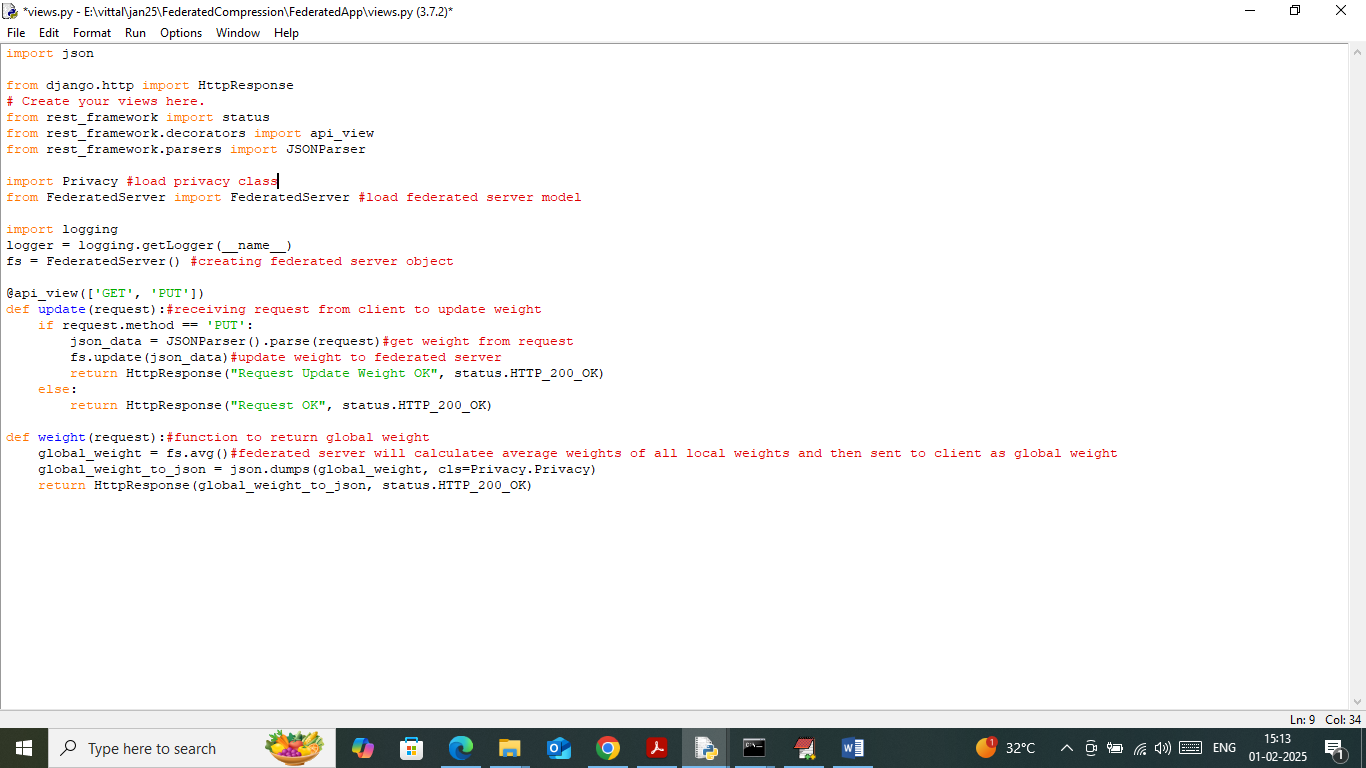
OMLSVD and auto-encoder required less storage image compare to existing algorithms. Traditional training of Auto-Encoder model require sharing of datasets or weights with all medical clients which may leak data of one medical client with other medical client.

To overcome from above data privacy and leakage we are employing Federated Learning technique where all clients will trained model on their datasets and then send their model local weights to Global federated server. Anytime any medical client can request federated server to get global weights of all clients and then perform prediction. So by employing Federated server we can get global dataset weight without publishing and leaking dataset so privacy will be achieved.

To train and test existing and propose auto-encoder algorithms we have used 3D chest dataset which can be download from below link

<https://www.kaggle.com/datasets/constantinseibold/anatomy-in-chest-x-rays-pax-ray>

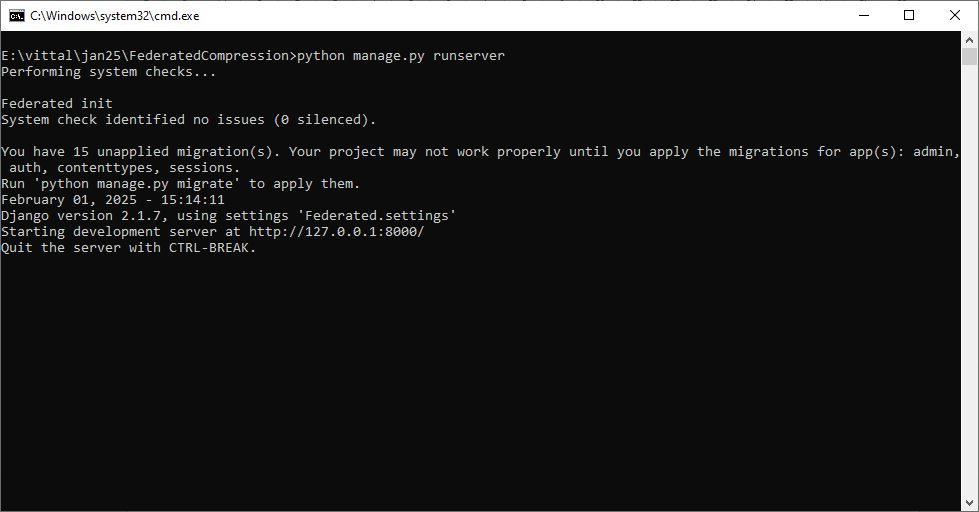
We have designed following Federated Server to manage local and global weights



In above screen read red colour comments to know about local and global model weights of Federated server with privacy.

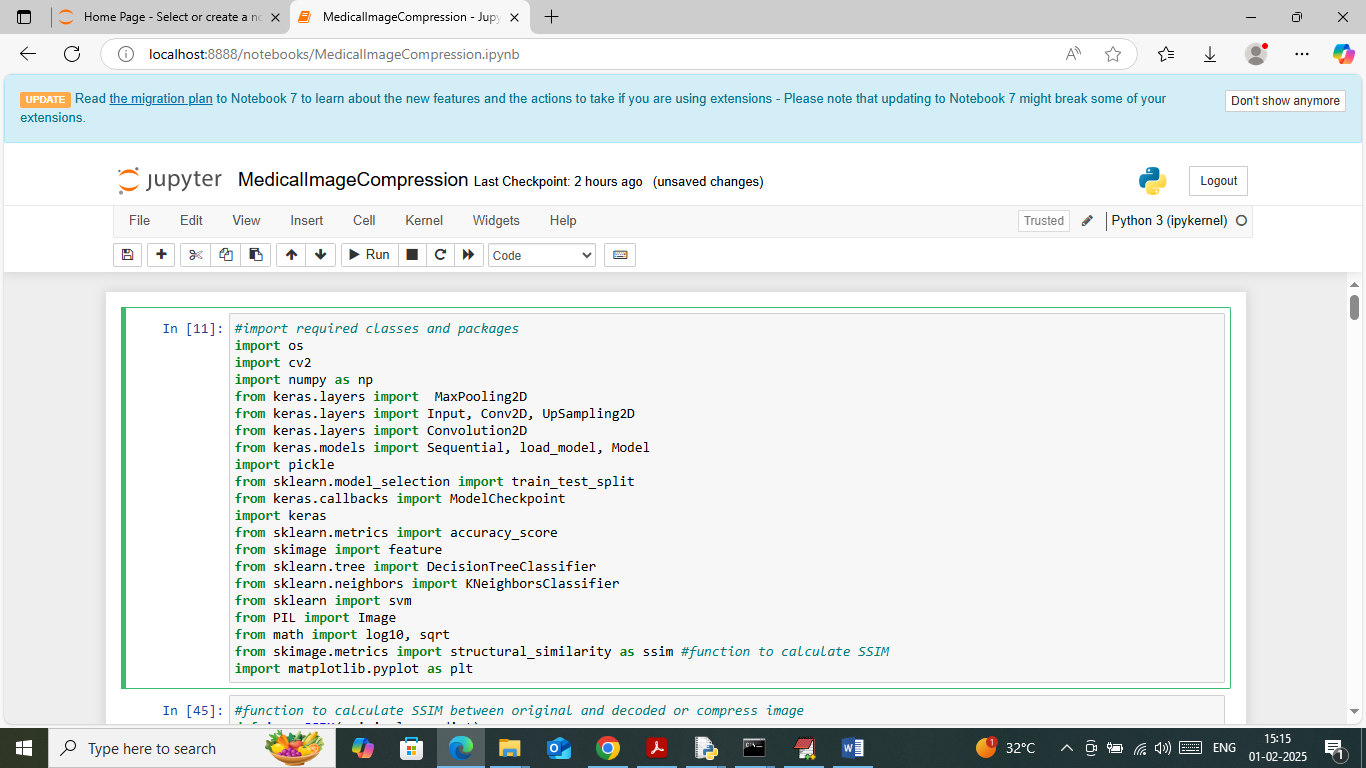
SCREEN SHOTS

First double click on ‘runFLServer.bat’ file to start federated server and then will get below page

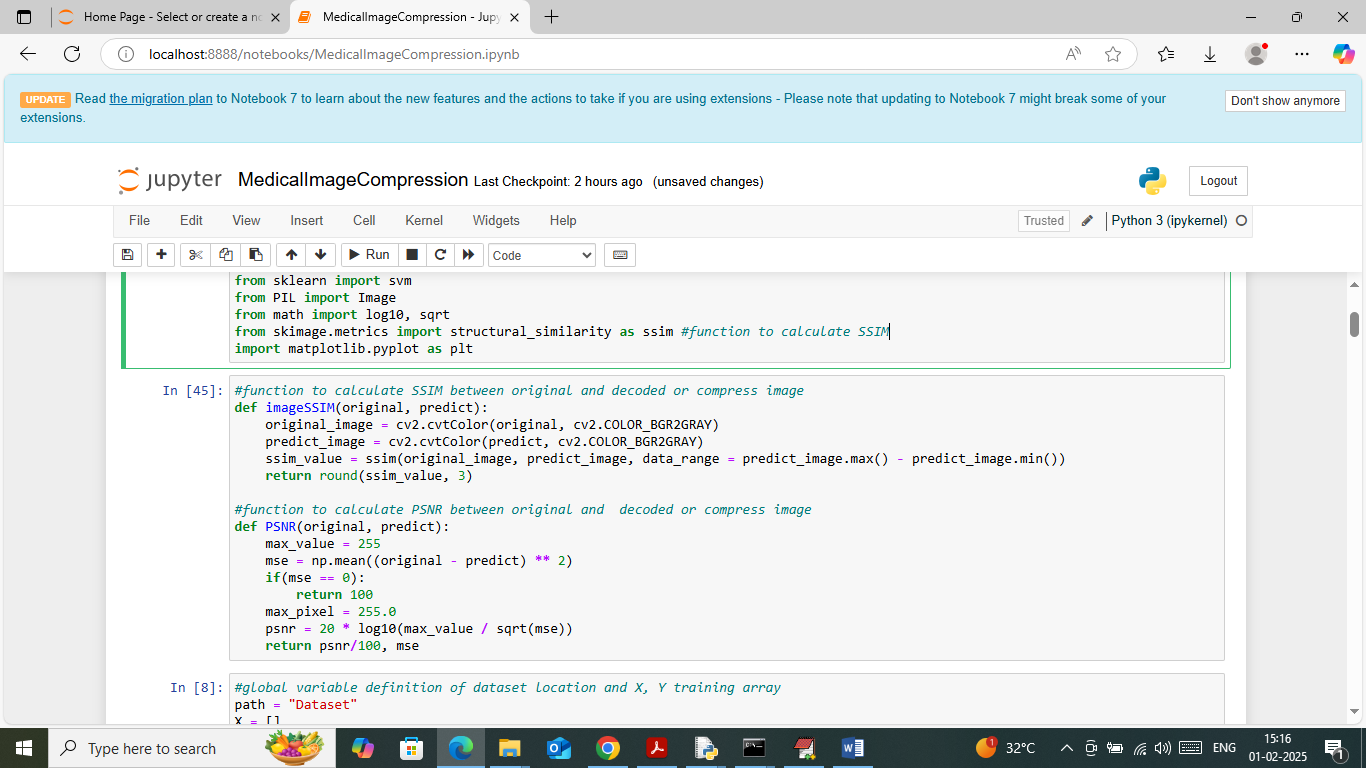


In above screen Federated server started and now double click on ‘runJupyter.bat’ file to start JUPYTER notebook and then can see below output screens.

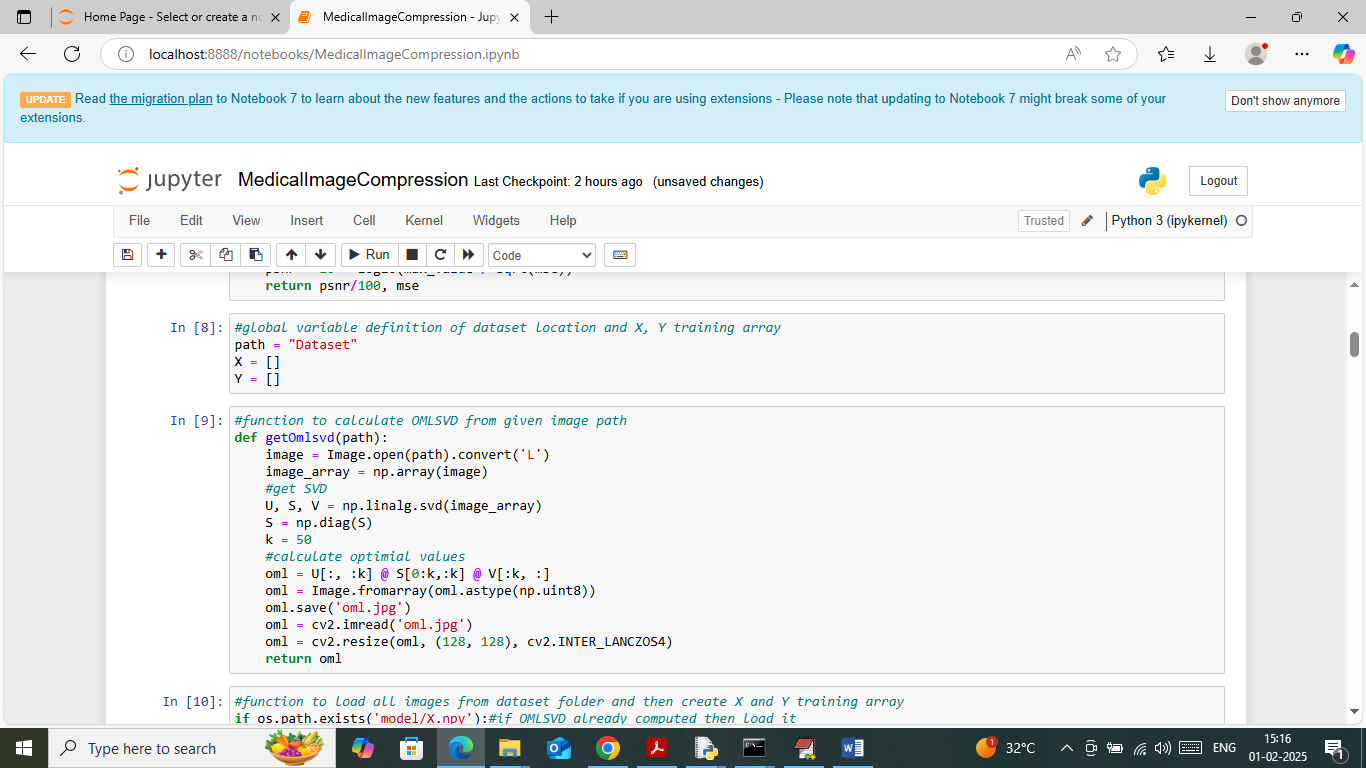
We have coded this project using JUPYTER notebook and below are the code and output screens with blue colour comments



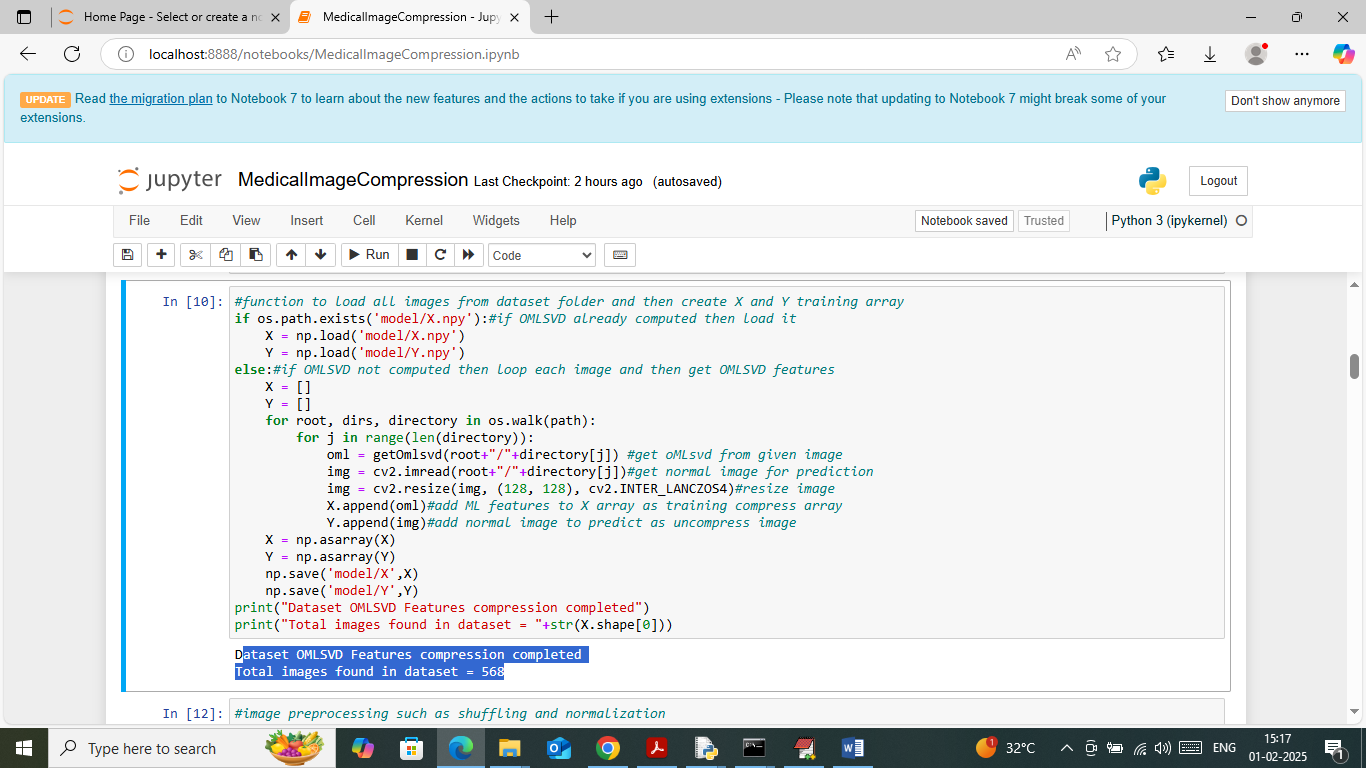
In above screen importing required python classes and packages



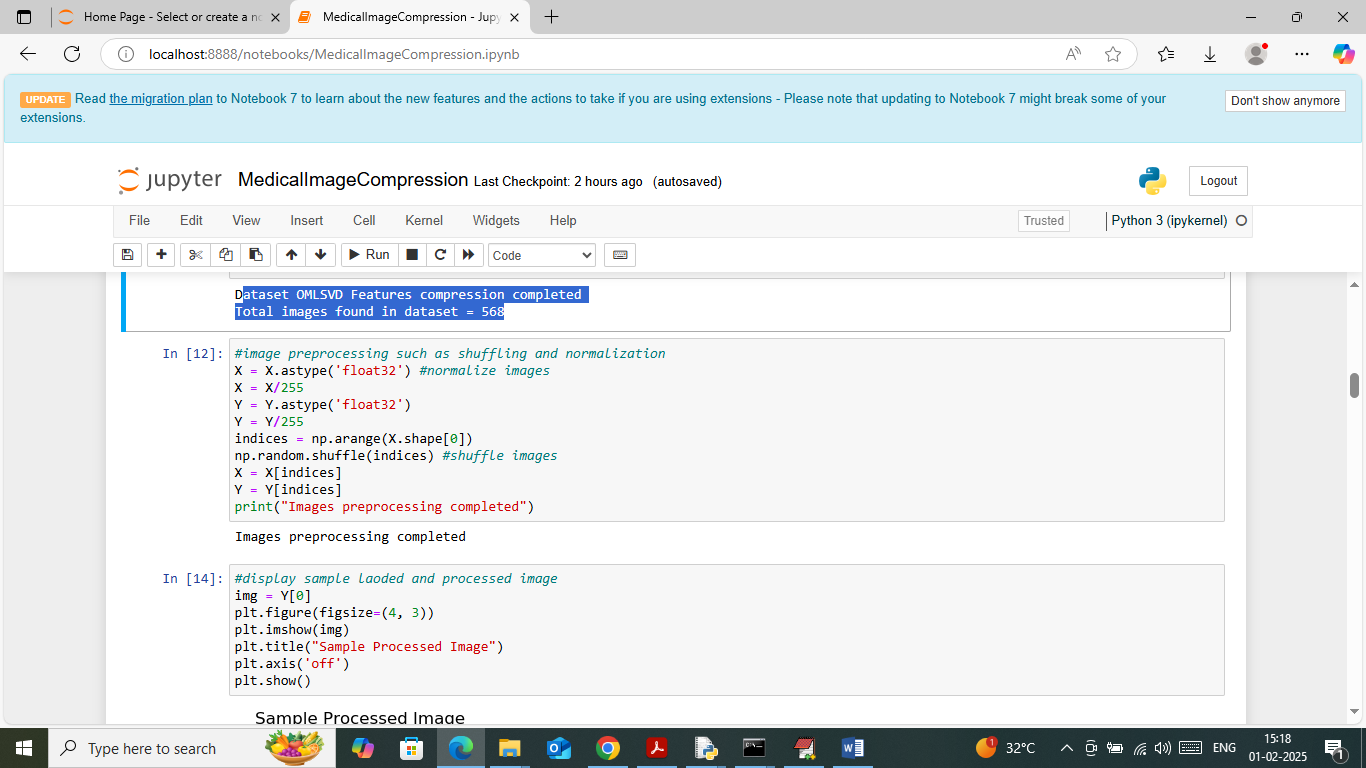
In above screen defining function to calculate SSIM and PSNR values



In above screen defining function to compress image using OMLSVD



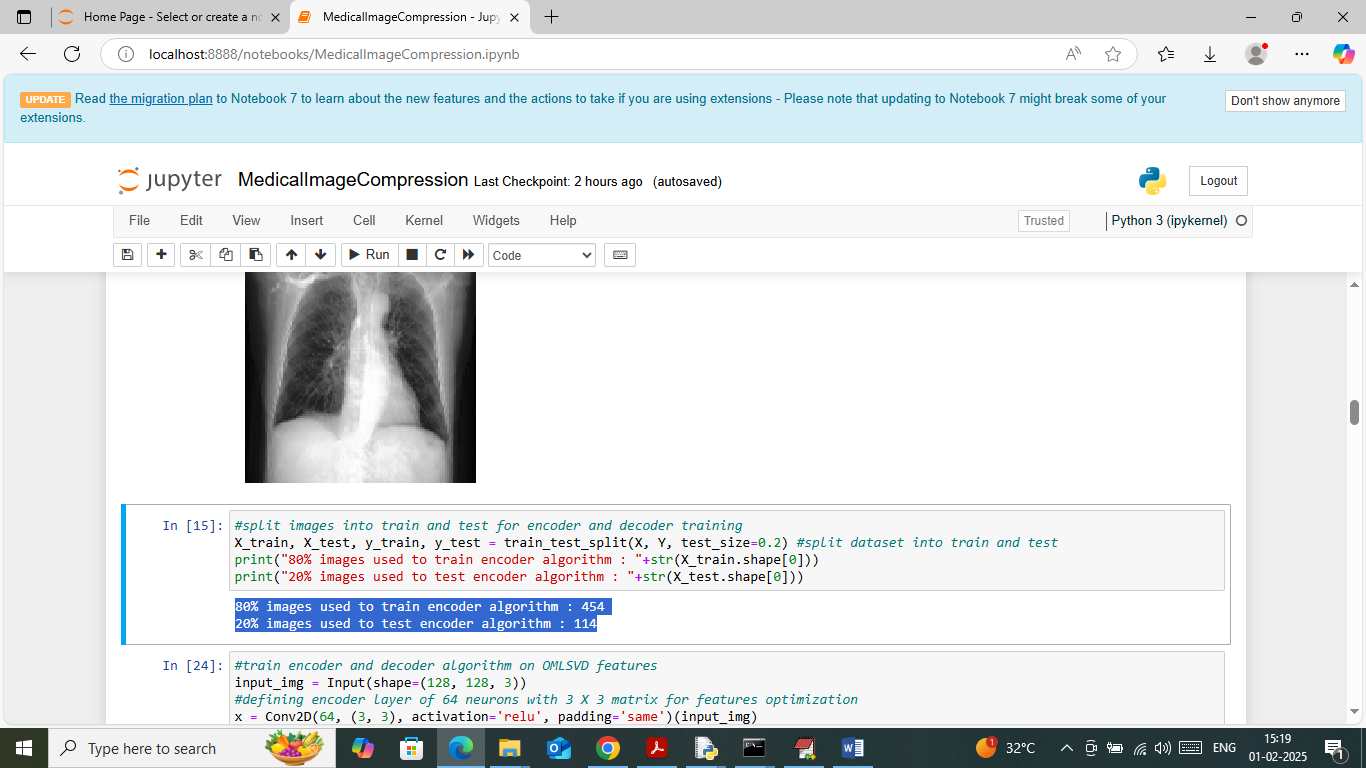
In above screen looping and reading all images from dataset and then compressing using OMLSVD and then creating training array. In above screen can total number of images loaded



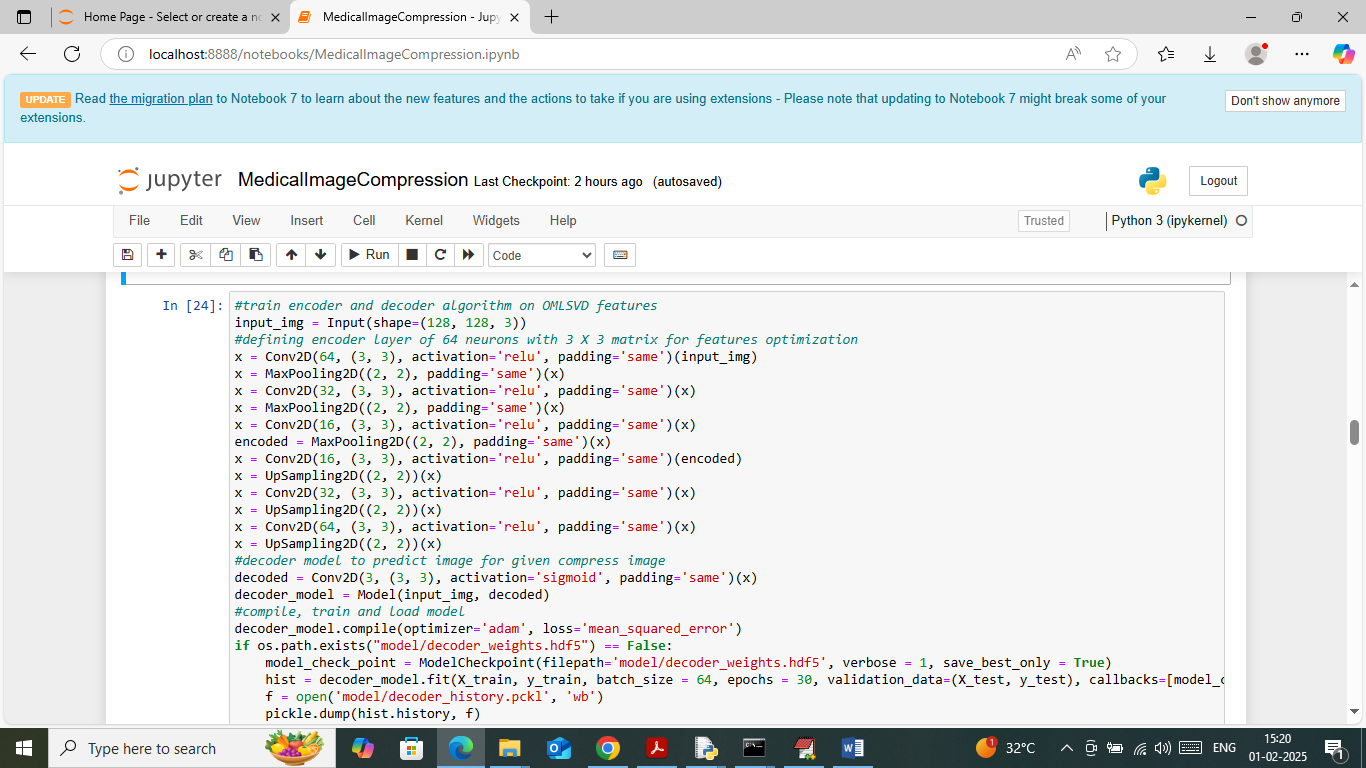
In above screen applying image processing technique to normalize and shuffle images



In above screen displaying sample processed image



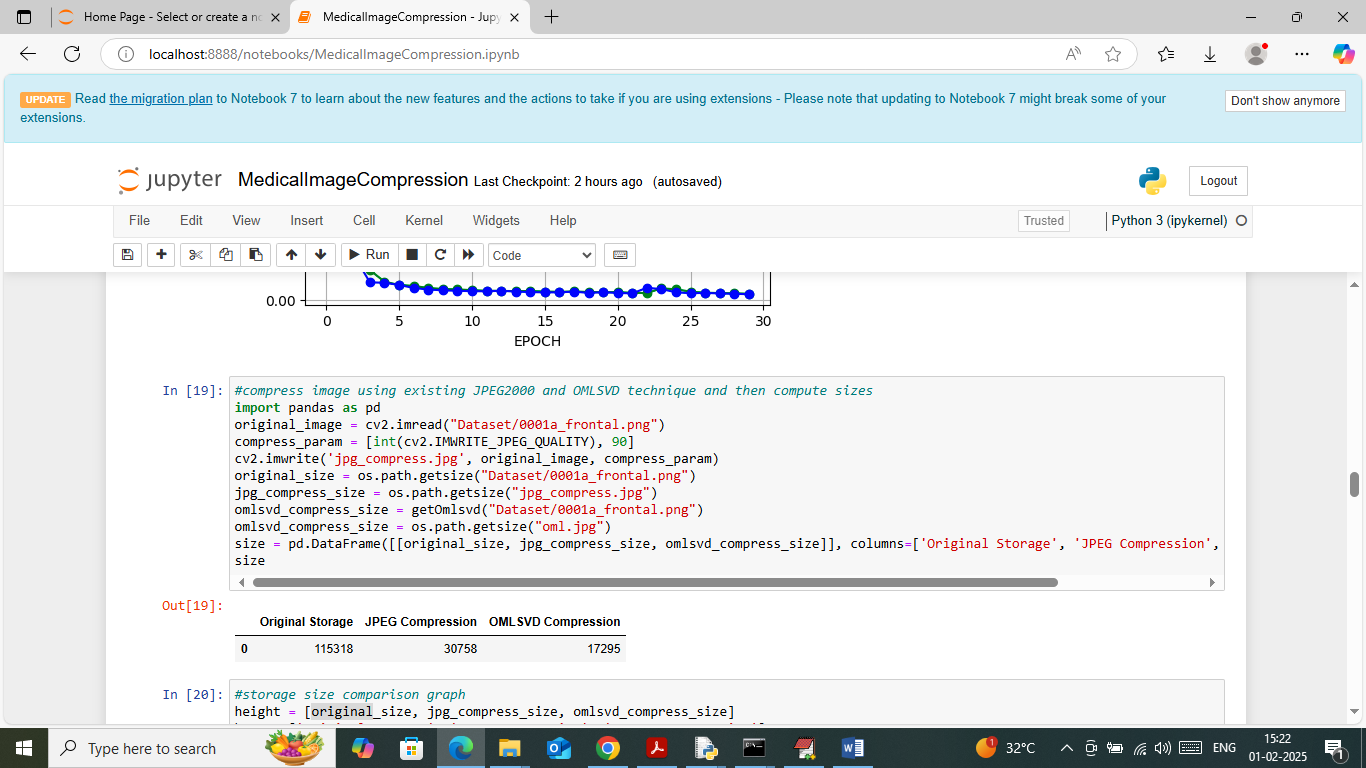
In above screen defining function to split dataset into train and test where application using 80% images for training and 20% for testing



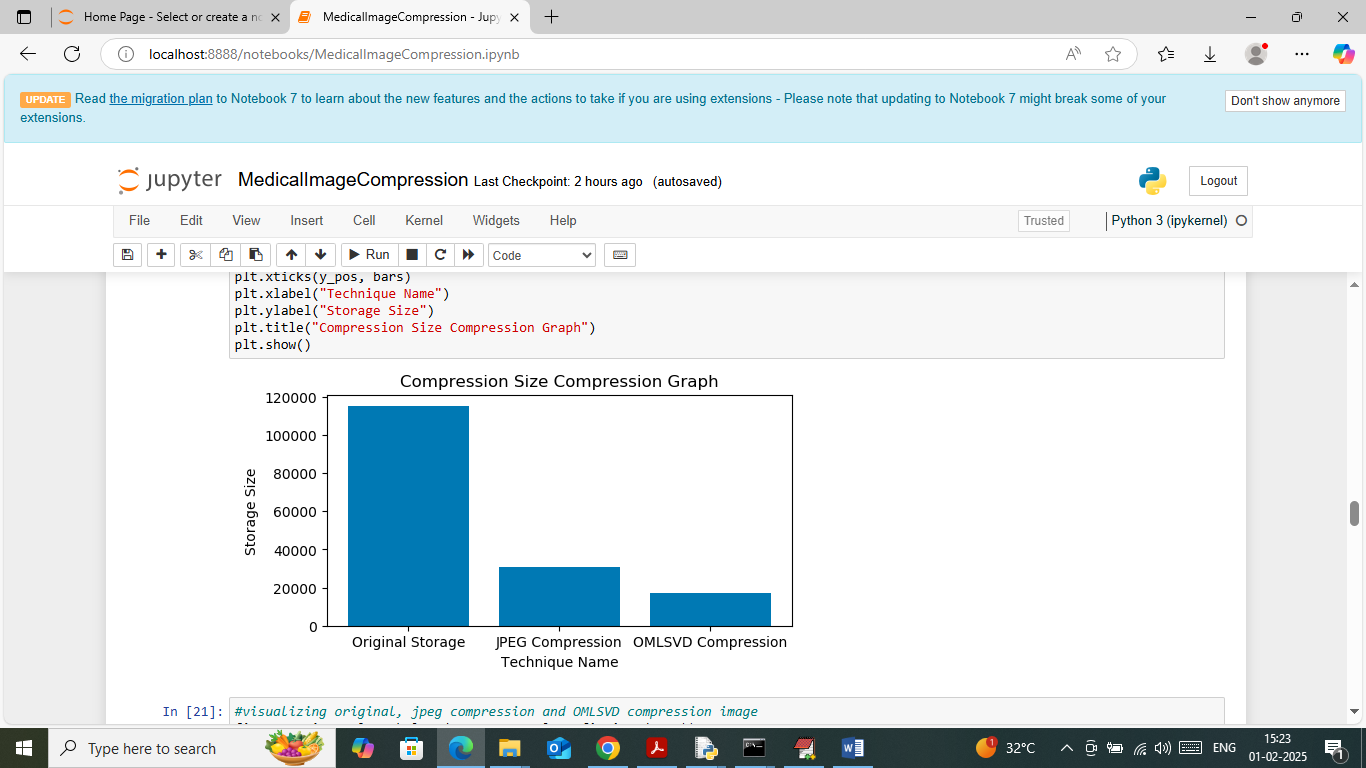
In above screen defining auto-encoder model to train compressed images and then will get below output



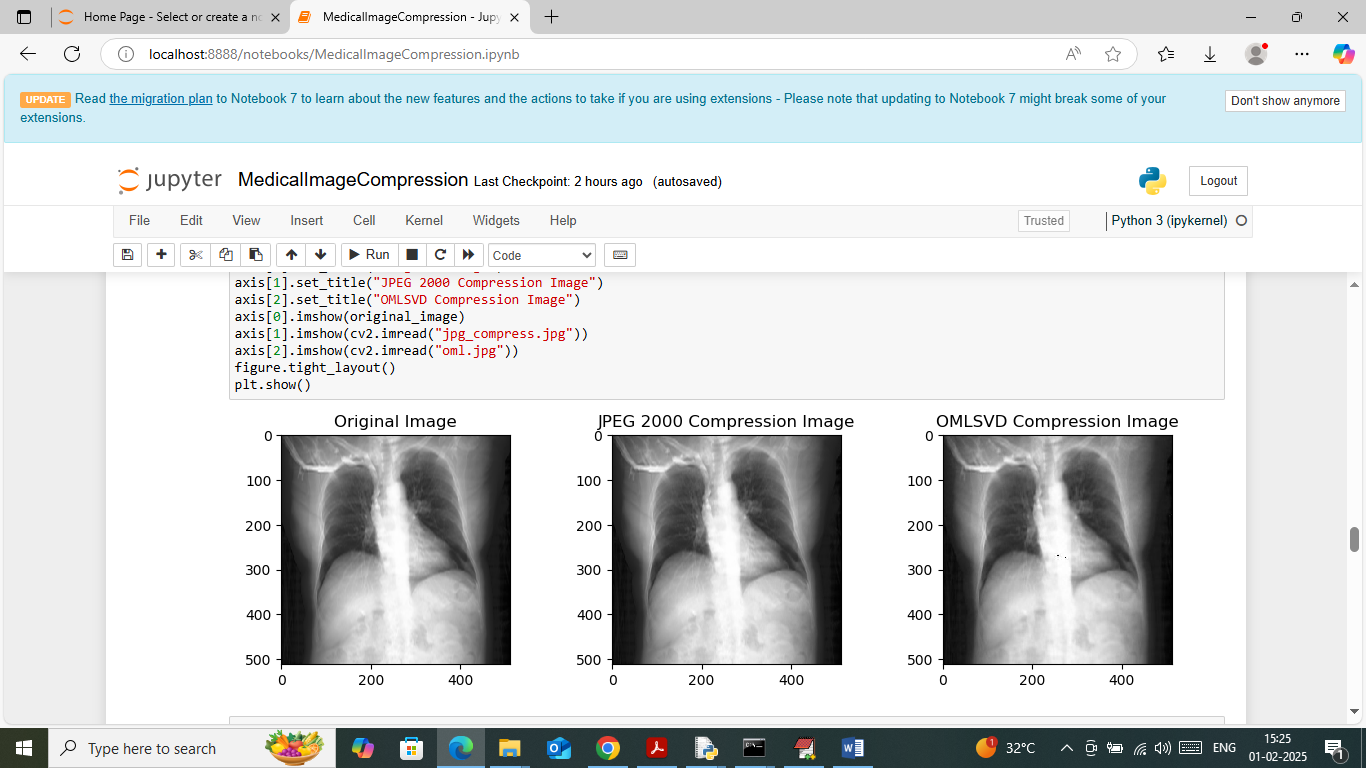
In above screen auto-encoder model training completed and in graph can see auto-encoder model training and validation loss. In above graph x-axis represents ‘Number of Epoch’ and y-axis represents loss and can see with each increasing epoch loss values got decreased and reached closer to 0.



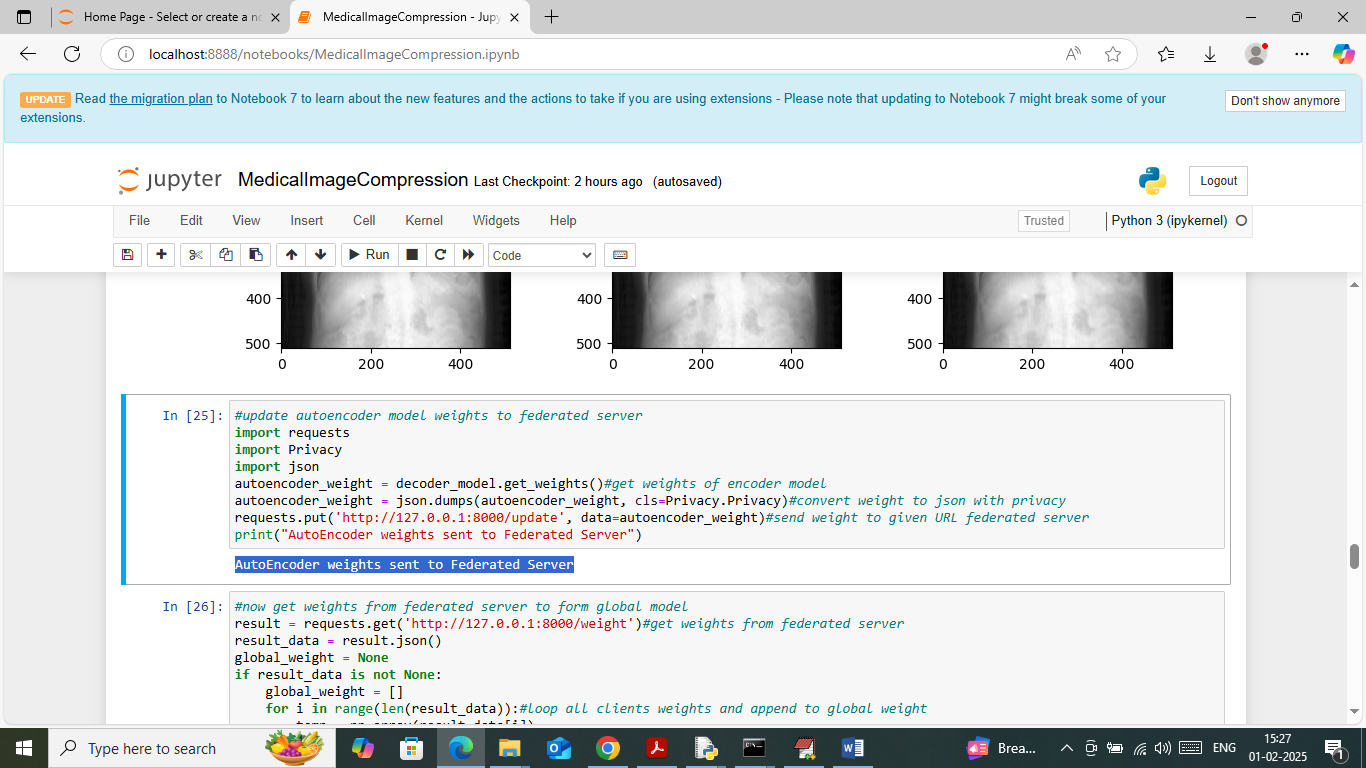
In above screen in tabular format can see original image size, JPEG2000 compress and OMMLSVD compress size and in all OMDSVL got less size



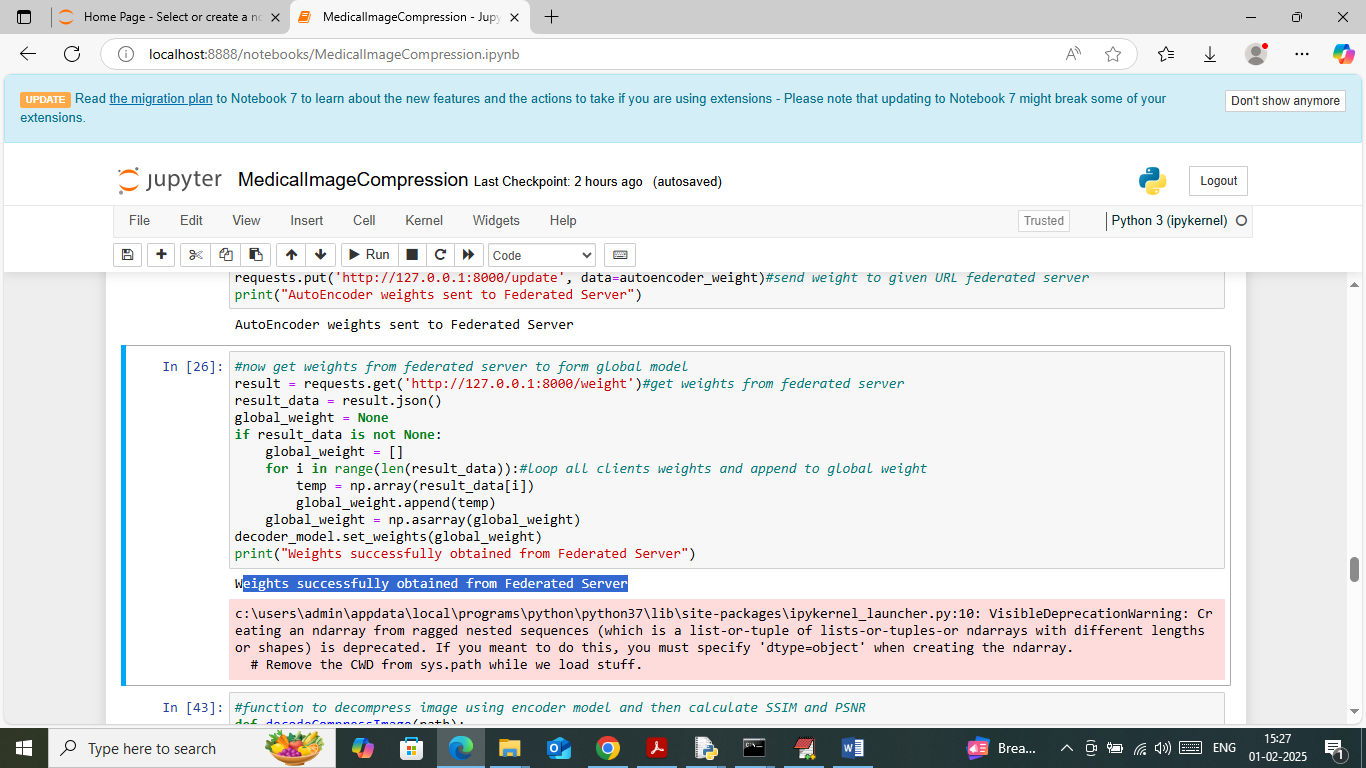
In above screen visualizing storage size comparison graph where x-axis represents compression type and y-axis represents size and in all propose OMLSVD got less storage size



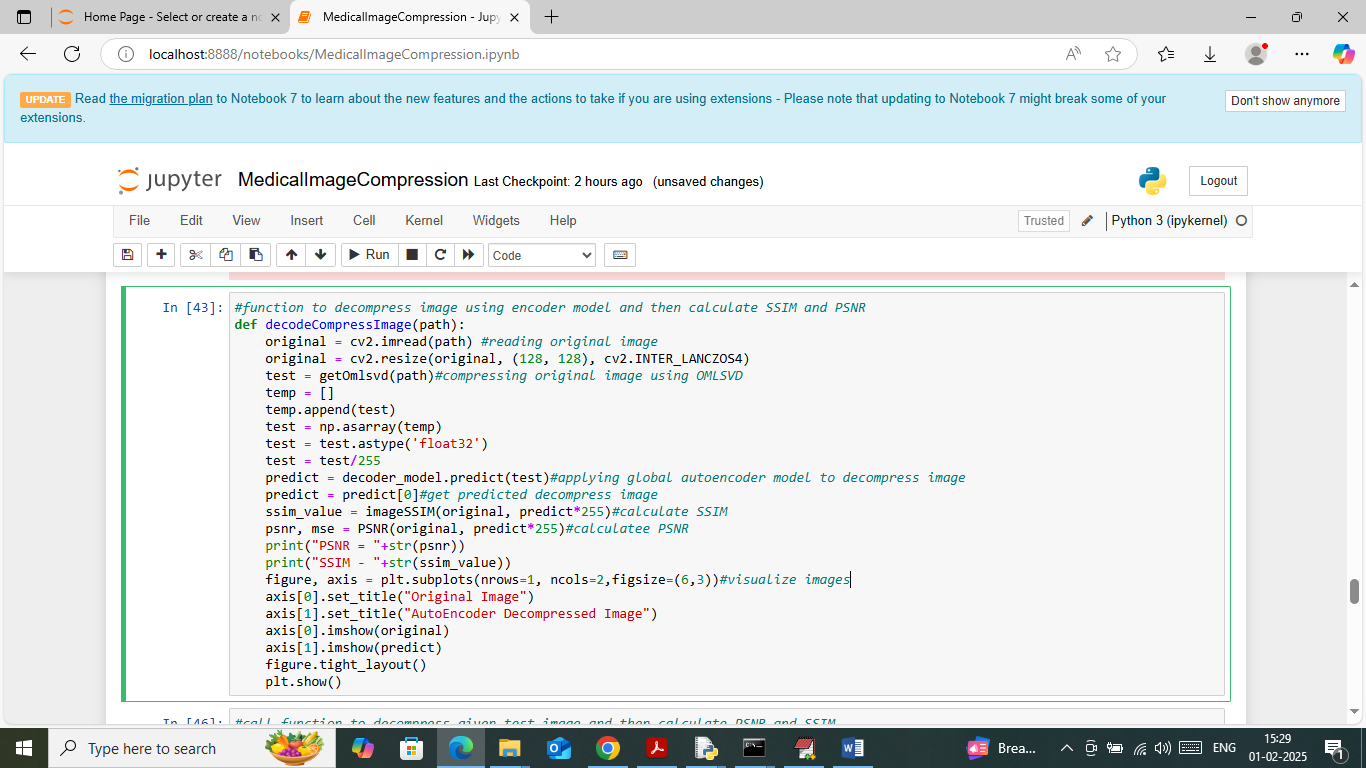
In above screen visualizing images where first image is the original image and second image is the JPEG2000 compress size and 3rd image is the OMLSVD compress image. In above screen can see all images look similar without any difference but we can see difference in storage size



In above screen uploading auto-encoder model to federated server by using federated server URL



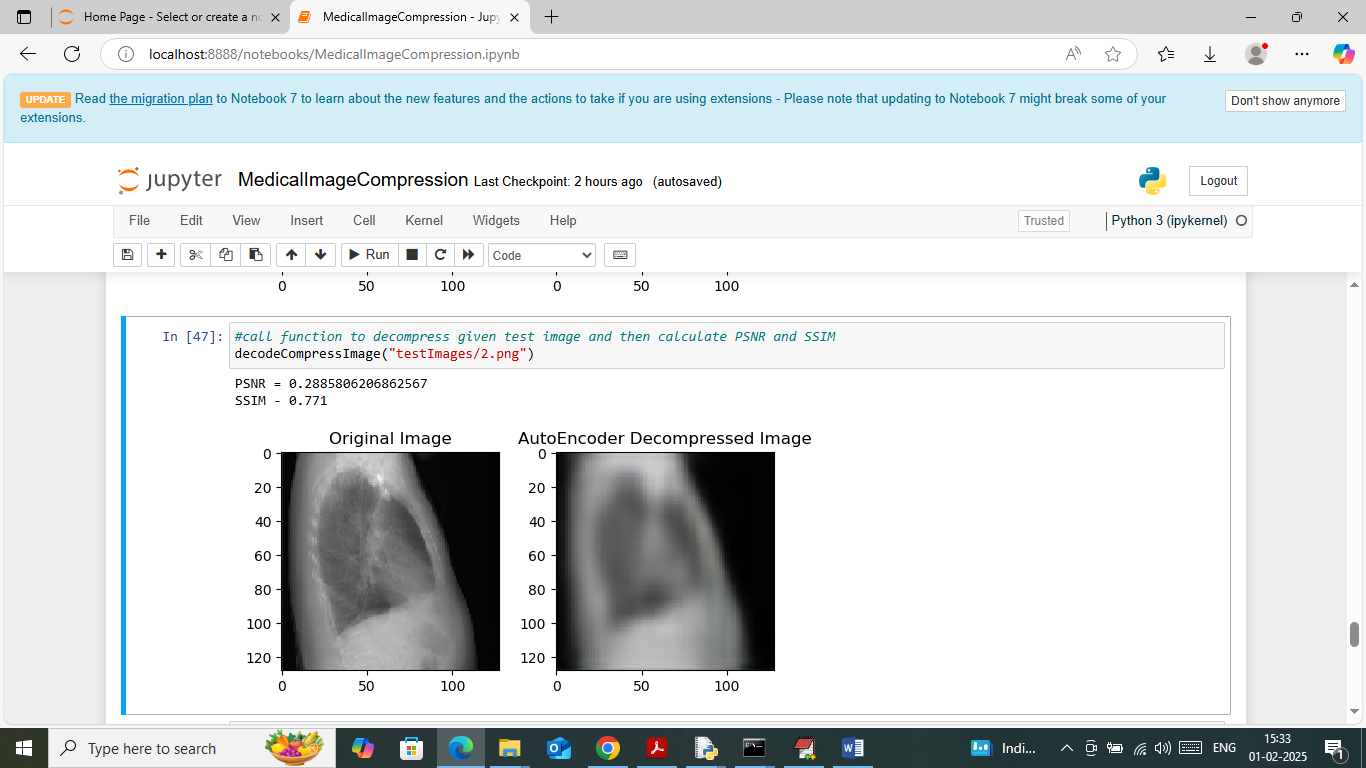
In above screen connecting to Federated Server and getting all weights as global weights



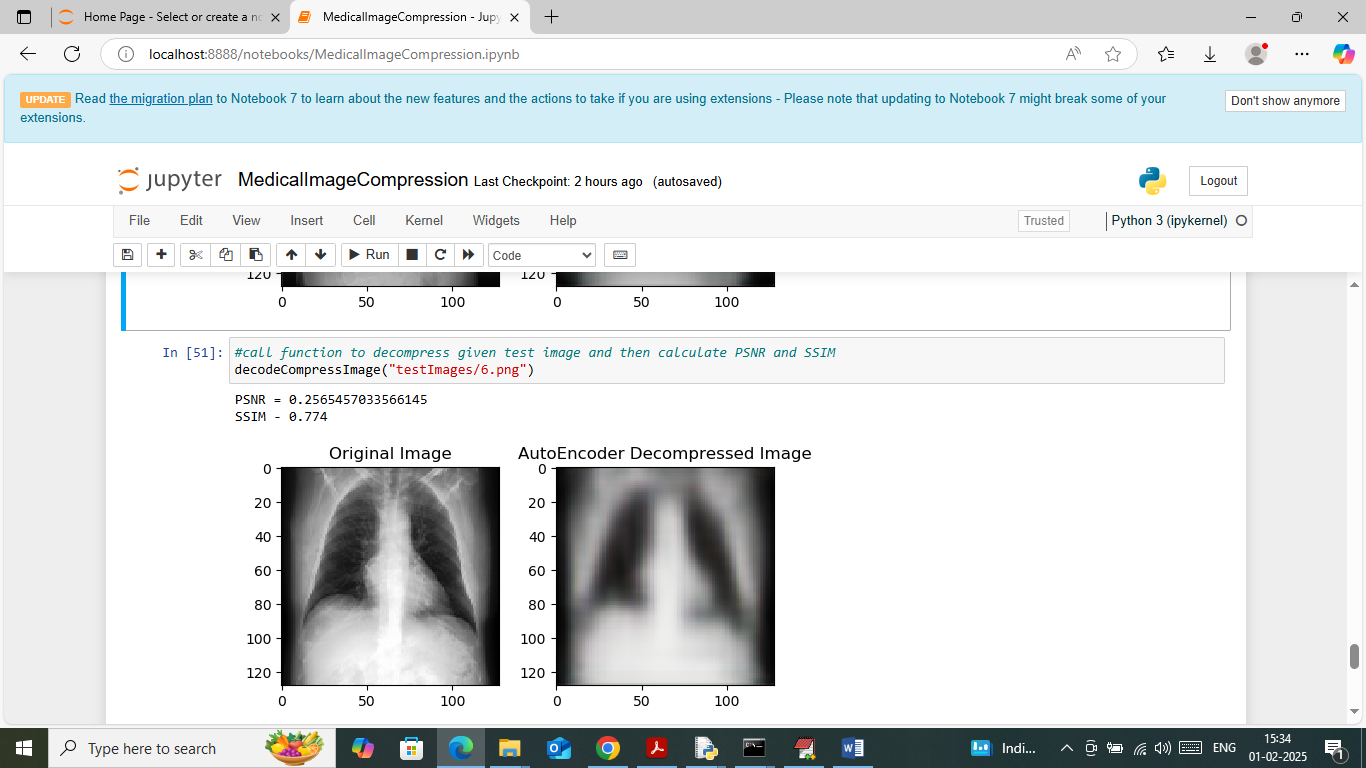
In above screen defining ‘Decoded decompress image’ function to dcompress image



In above screen calling decompress function with test image and then first image is the original image and second image is the auto-encoder decompress image which are looking same. In blue text can see SSIM and PSNR values where SSIM (structural similarity image) must be closer to 1 and PSNR (peak signal noise ratio) must be closer to 0.



In above screen predicting pr decompress another image out[ut where original and decompress images are same



In above screen can see another image output.

Similarly change test image path and predict decompress image for any other test image.

**Conclusion:**

The proposed federated learning-based 3D medical image compression system presents a novel and effective solution to the challenges of privacy, data security, and scalability faced by traditional centralized compression methods. By keeping patient data localized and only sharing encrypted model updates, it ensures strict compliance with healthcare regulations such as HIPAA and GDPR. This approach not only protects sensitive medical information but also reduces the communication overhead typically associated with transmitting large imaging datasets. The use of advanced deep learning models like 3D convolutional autoencoders allows the system to achieve high compression ratios while preserving essential diagnostic details, making it suitable for clinical use.

Furthermore, the decentralized federated architecture enhances system robustness by eliminating single points of failure and enabling seamless integration of multiple medical institutions. Techniques such as differential privacy and secure aggregation safeguard against potential privacy leaks during model updates, fostering trust among participants. This scalable and secure framework facilitates collaborative AI model development across diverse healthcare environments, including resource-constrained and rural settings. Overall, the system paves the way for more efficient, secure, and cooperative management of medical imaging data, ultimately supporting improved patient care and advancing medical research.

**Future Work:**

Although the proposed federated learning-based 3D medical image compression system shows significant promise, there are several important areas for future improvement. A key challenge in federated learning is handling the heterogeneity of data across different medical institutions. Since patient demographics, imaging devices, and protocols vary widely, the data distribution is often non-identical and non-independent (non-IID). This can negatively impact the convergence and accuracy of the global compression model. Future work could focus on personalized federated learning approaches, where each client fine-tunes the global model locally to better fit its specific data characteristics. This personalization would improve compression quality and diagnostic reliability for each participating site while maintaining the benefits of collaborative training.

Another important area of future research is improving communication efficiency during federated training. Although the system reduces the need to transmit large raw datasets, frequent sharing of model updates can still impose significant bandwidth demands, especially in institutions with limited network resources. Adaptive communication protocols could be developed to dynamically adjust update frequency and size based on current network conditions and model training progress. Additionally, exploring more compact model architectures or pruning techniques would help reduce the size of transmitted updates, thereby minimizing communication overhead without sacrificing compression performance.

The integration of edge computing with federated learning also presents a promising direction for future development. By leveraging edge devices at local institutions to perform preliminary compression and inference tasks, the system can reduce latency and offload some computational burdens from central servers. Cloud infrastructure can then focus on aggregating model updates and distributing the global model. This hybrid edge-cloud approach would enhance responsiveness, particularly in real-time clinical scenarios, and improve scalability by balancing the computational load more effectively.

Security and privacy remain paramount in medical applications, and future work should continue to strengthen these aspects. Beyond the current use of differential privacy and secure aggregation, more advanced cryptographic methods like fully homomorphic encryption or blockchain-based audit trails could be explored to provide stronger protection against data leakage and tampering. Moreover, developing mechanisms to detect and defend against malicious clients or adversarial attacks within the federated network will be crucial to preserving model integrity and trust.

Finally, broad clinical validation is necessary to demonstrate the system’s effectiveness in real-world settings. Future studies should evaluate the model on diverse, multi-institutional datasets, potentially incorporating multi-modal data such as electronic health records alongside medical images. This will help assess its impact on clinical workflows, diagnostic accuracy, and data management efficiency. Expanding the system to support compression of other medical data types will also increase its versatility and usefulness across various healthcare domains.

**References**

1. McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS), 54, 1273–1282.
2. Sheller, M. J., Reina, G. A., Edwards, B., Martin, J., & Bakas, S. (2020). Federated Learning in Medicine: Facilitating Multi-Institutional Collaborations Without Sharing Patient Data. Scientific Reports, 10(1), 12598.
3. Zhang, J., Xie, Y., Wang, Y., & Xia, Y. (2021). Deep Learning-Based Medical Image Compression: A Review. IEEE Access, 9, 48170–48191.
4. Kaul, R., Mishra, A., & Nain, N. (2020). 3D Convolutional Autoencoder for Medical Image Compression. IEEE International Conference on Computer Vision Workshop (ICCVW), 2877–2886.
5. Kairouz, P., McMahan, H. B., et al. (2019). Advances and Open Problems in Federated Learning. arXiv preprint arXiv:1912.04977.
6. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., ... & Seth, K. (2017). Practical Secure Aggregation for Privacy-Preserving Machine Learning. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 1175–1191.
7. Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., & Chan, K. (2019). Adaptive Federated Learning in Resource Constrained Edge Computing Systems. IEEE Journal on Selected Areas in Communications, 37(6), 1205–1221.
8. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
9. HIPAA Privacy Rule, U.S. Department of Health & Human Services. (1996). Available: https://www.hhs.gov/hipaa/index.html
10. GDPR: General Data Protection Regulation (EU) 2016/679. Available: <https://gdpr-info.eu/>