**Person Vein Identification using CNN**

**1.INTRODUCTION:**

Automated personal identification using unique anatomical characteristics of humans is widely employed for e-governance, border crossing security and a range of e-business applications. There has been significant increase in the detection of surgically altered fingerprints, fake iris stamps, or the usage of sophisticated face masks, during the last decade. Vascular biometrics identification, like using finger vein images, can help to preserve the integrity of biometrics system as it’s extremely difficult to surgically alter vascular biometrics. Another advantage with the usage of finger vein image based identification lies in the enhanced anonymity during personal authentication as the subsurface vascular patterns are largely hidden underneath and difficult to steal or imaged under visible illumination. The possibility of personal identification using vascular patterns imaged by the light transmitted through hands was indicated in 1992 but was not demonstrated until 2000 . Such earliest work demonstrated feasibility of finger vein identification using the normalized-cross correlation. Miura et al. later introduced repeated line tracking approach to improve the performance of finger vein identification, and they further enhanced the performance with maximum curvature . Kumar and Zhou introduced earliest publicly accessible finger-vein images database and comparatively evaluated range the effectiveness of hand-crafted features for the finger-vein identification. The method introduced in using Gabor filter based enhancement and morphological operations is still regarded the best performing methods for matching finger-vein images. A range of other hand-crafted finger vein features primarily obtained from the careful evaluation of the registered images, have been attempted in the literature with very promising results. Multiple features acquired from the two cameras or using multiple feature extractors can be combined to significantly improve the performance for the vascular biometrics matching. One of the limitations of finger-vein identification methods introduced in the literature is related to their large template size. Smaller template size is highly desirable to reduce storage and/or enhance the matching speed for the online applications. There have also been successful attempts to reduce the finger-vein template size, like in or recently in using sparse representation of enhanced.

* 1. **Objective of the project:**

Automated personal identification using vascular biometrics, such as from the finger vein images, is highly desirable as it helps to protect the personal privacy and anonymity in automated personal identification. The Convolutional Neural Network (CNN) has shown remarkable capability for learning biometric features that can offer robust and accurate matching. This paper introduces a new approach for the finger vein authentication using the CNN and supervised discrete hashing. We also systematically investigate comparative performance using several popular CNN architectures in other domains, i.e., Light CNN, VGG-16, Siamese and the CNN with Bayesian inference based matching. The experimental results are presented using a publicly available two-session finger-vein database. Most accurate performance is achieved by incorporating supervised discrete hashing from a CNN trained using the triplet-based loss function. The proposed approach not only achieves outperforming results over other considered CNN architecture available in the literature but also offers significantly reduced template size as compared with those over the other finger-vein images matching methods available in the literature.

**2. LITERATURE SURVEY:**

**“An efficient finger-vein extraction algorithm based on random forest regression with efficient local binary patterns.”**

Finger-vein, as a secure and convenient biometric characteristic in nature, has been widely studied for authentication in recent years. In this paper, we propose an efficient finger-vein extraction algorithm based on random forest training and regression with efficient local binary pattern feature. By integrating with a vein pattern matching method which is robust to finger misalignment, we achieved state-of-the-art finger-vein recognition. Thorough experiments have been conducted on two popular databases to prove the effectiveness and robustness of the proposed method.

“**Fusion of band limited phase only correlation and width centroid contour distance for finger based biometrics.”**

This research presents a novel method of multimodal finger biometrics that combines the identification of finger vein and finger shape. The suggested technique uses Band Limited Phase Only Correlation (BLPOC) to gauge how similar two pictures of finger veins are. Unlike earlier techniques, BLPOC can improve finger vein detection performance since it is resistant to noise, occlusions, and rescaling effects. Regarding the identification of finger geometry, a novel class of geometric properties known as Width-Centroid Contour Distance (WCCD) is put forth. The finger width and Centroid Contour Distance (CCD) are combined in this WCCD. Finger geometry recognition accuracy can be increased by fusing W and CCD features together as opposed to using only one type of feature. Ultimately, we utilize a score-level fusion method based on the weighted SUM rule to merge the recognitions of the finger vein and finger shape. Using our own database of 123 volunteers, we conducted an experimental evaluation that produced an effective recognition performance with an equal error rate (EER) of 1.78% and a total processing time of 24.22 ms.

**“Finger-vein biometric identification using convolutional neural network.”**

The use of human finger-vein traits for the purpose of automatic user recognition has gained a lot of attention in recent years. Current state-of-the-art techniques can provide relatively good performance, yet they are strongly dependent upon the quality of the analyzed finger-vein images. In this paper, we propose a convolutional-neural-network-based finger-vein identification system and investigate the capabilities of the designed network over four publicly available databases. The main purpose of this paper is to propose a deep-learning method for finger-vein identification, which is able to achieve stable and highly accurate performance when dealing with finger-vein images of different quality. The reported extensive set of experiments show that the accuracy achievable with the proposed approach can go beyond 95% correct identification rate for all the four considered publicly available databases.

**“A new system for human authentication using finger vein images.”**

A new system (ARTeM) for human authentication using finger vein images is described here. The developed algorithm combines 1) a fuzzy contrast enhancement algorithm with 2) a mutual information and affine transformation based registration technique and 3) a correlation coefficient based template matching algorithm, to detect the identity of a person based on the match-scores with finger vein images stored in the database. For performance assessment of the ARTeM algorithm, the benchmark SDUMLA multimodal biometric database containing 3816 images of 106 persons is used. On the complete database, up to 95.28% classification accuracy is achieved with single finger images; while up to 98.11% accuracy is observed with a consensus of two fingers. On a reduced subset of 86 persons’ database, 98.84% accuracy is achieved with single finger classification and cent percent classification is obtained using a consensus of two fingers. Comparative analyses with other works also validate the effectiveness of the developed methodology.

**“Vascular pattern analysis towards pervasive palm vein authentication.”**

In this paper we propose an Image Analysis technique for Vascular Pattern of Hand Palm, which in turn leads towards Palm Vein Authentication of an individual. Near-Infrared Image of Palm Vein pattern is taken and passed through three different processes or algorithms to process the Infrared Image in such a way that the future authentication can be done accurately or almost exactly. These three different processes are: a. Vascular Pattern Marker Algorithm (VPMA); b. Vascular Pattern Extractor Algorithm (VPEA); and c. Vascular Pattern Thinning Algorithm (VPTA). The resultant Images will be stored in a Database, as the vascular patterns are unique to each individual, so future authentication can be done by comparing the pattern of veins in the palm of a person being authenticated with a pattern stored in a database.

**“Enhancement of finger-vein image by vein line tracking and adaptive gabor filtering for finger-vein recognition.”**

Biometrics is the technology to identify a user by using the physiological or behavioral characteristics. Among the biometrics such as fingerprint, face, iris, and speaker recognition, finger-vein recognition has been widely used in various applications such as door access control, financial security, and user authentication of personal computer, due to its advantages such as small sized and low cost device, and difficulty of making fake vein image. Generally, a finger-vein system uses near-infrared (NIR) light illuminator and camera to acquire finger-vein images. However, it is difficult to obtain distinctive and clear finger-vein image due to skin scattering of illumination since the finger-vein exists inside of a finger. To solve these problems, we propose a new method of enhancing the quality of finger-vein image. This research is novel in the following three ways compared to previous works. First, the finger-vein lines of an input image are discriminated from the skin area by using local binarization, morphological operation, thinning and line tracing. Second, the direction of vein line is estimated based on the discriminated finger-vein line. And the thickness of finger-vein in an image is also estimated by considering both the discriminated finger-vein line and the corresponding position of finger-vein region in an original image. Third, the distinctiveness of finger-vein region in the original image is enhanced by applying an adaptive Gabor filter optimized to the measured direction and thickness of finger-vein area. Experimental results showed that the distinctiveness and consequent quality of finger-vein image are enhanced compared to that without the proposed method.

**“A low cost finger vein authentication system, using maximum curvature points, in: International Conference on Applied Electronics.”**

Finger vein patterns have been proposed as a suitable biometric feature for authentication applications. Systems using this feature are generally low cost, accurate and easy to use. Such a system is described in this work. Infrared light is used to capture an image of a finger and a pattern recognition algorithm extracts the vein patterns. For robust and precise extraction of the depicted veins, a method of calculating local maximum curvatures in cross-sectional profiles of a vein image was used. The authentication algorithm has been evaluated with images acquired from the device built, as well as with images from a 400 fingers database. The false acceptance and false rejection rates obtained were promising.

**“Finger vein recognition using mutual sparse representation classification. IET Biometrics.”**

Sparse representation classification (SRC) is one of the popular methods of classification in biometrics, in which the decision of class for the test sample was based on the class with minimum reconstruction error. As SRC is based on the sparsity of the images, a decision based on reconstruction error is not ideal. In this study, an efficient classification methodology for finger vein recognition called mutual SRC (MSRC) is proposed. MSRC classifies the test sample by a new decision rule which significantly improves the recognition rate of the conventional SRC. By this new decision rule, the classification of the test sample is not only based on the nearest sparse neighbour but also on determining the training sample which considers the test sample as its nearest neighbour (NN). In this method, the training set is selected based on reconstruction error for the test sample, then which training sample considers the test sample as its NN is identified by sparse representation. Increases of 4.67, 10.59, 26.82, and 3.44% in the recognition rates are observed for the proposed MSRC method when compared with conventional SRC using the four public finger vein database.

**3. SYSTEM ANALYSIS**

**3.1 Existing System**

All existing biometric algorithms are dependent on humans hand crafted features whose prediction accuracy is not up to the mark.

**Disadvantages**

1. It takes more time
2. Less accuracy

**3.2 Proposed System**

In propose work we are employing machine learning SVM algorithm and deep learning CNN algorithm and then evaluating both performance in terms of accuracy and confusion matrix.

**Advantages**

1.It takes less time

2.More prediction

**Modules:**

To implement this project we have designed following modules

1. Upload Finger Vein Dataset: using this module we will upload dataset to application and then find and plot different person fingers found in dataset
2. Preprocess Dataset: using this module we will read image and then resize all images to equal size and then shuffle and normalize images and then split dataset into train and test where application using 80% images for training and 20% for testing
3. Run SVM Algorithm: 80% processed trained images will be input to SVM algorithm to train a model and then 20% test images will be applied on trained model to calculate prediction accuracy
4. Run CNN Algorithm: 80% processed trained images will be input to CNN algorithm to train a model and then 20% test images will be applied on trained model to calculate prediction accuracy
5. Comparison Graph: using this module we plot comparison graph between both algorithms
6. Identify Finger Vein from Test Image: using this module we will upload test finger vein image and then CNN will analyse vein image and then identify person from that image.

**3.3. PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Umbrella Model):**

**Umbrella Activity**

**Umbrella Activity**

**Umbrella Activity**

1. Feasibility Study
2. TEAM FORMATION
3. Project Specification PREPARATION

Business Requirement Documentation

ANALYSIS & DESIGN

CODE

UNIT TEST

DOCUMENT CONTROL

ASSESSMENT

TRAINING

INTEGRATION & SYSTEM TESTING

DELIVERY/INSTALLATION

ACCEPTANCE TEST

Requirements Gathering

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

  
When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artefacts will be produced. Software artefacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artefacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artefact is linked to a specific design element, and that each developed artefact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artefacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artefacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) – a [requirements specification](http://en.wikipedia.org/wiki/Requirements_specification) for a [software system](http://en.wikipedia.org/wiki/Software_system) is a complete description of the behavior of a system to be developed. It includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. [Non-functional requirements](http://en.wikipedia.org/wiki/Non-functional_requirements) are requirements which impose constraints on the design or implementation (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_%28business%29) standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A [business analyst](http://en.wikipedia.org/wiki/Business_analyst), sometimes titled [system analyst](http://en.wikipedia.org/wiki/System_analyst), is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the [systems development lifecycle](http://en.wikipedia.org/wiki/Systems_development_life_cycle) domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

* [Business requirements](http://en.wikipedia.org/wiki/Business_requirements) describe in business terms what must be delivered or accomplished to provide value.
* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify .Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:
* **ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

* **Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

* **TECHNICAL FEASIBILITY**

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to .the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

**3.4.2. External Interface Requirements**

**User Interface**

The user interface of this system is a user friendly python Graphical User Interface.

**Hardware Interfaces**

The interaction between the user and the console is achieved through python capabilities.

**Software Interfaces**

The required software is python.

**HARDWARE REQUIREMENTS:**

# Processor - I3(min)

* Speed - 1.1 GHz
* RAM - 4GB(min)
* Hard Disk - 500GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - SVGA

**SOFTWARE REQUIREMENTS:**

* Operating System - Windows 10/above
* Programming Language - Python 3.7

**4. SYSTEM DESIGN**

**UML Diagram:**

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + **User Model View**
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + **Structural Model view**
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* **Behavioural Model View**

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* **Implementation Model View**

In this the structural and behavioural as parts of the system are represented as they are to be built.

* **Environmental Model View**

In this the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

**Class Diagram:**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

* The upper part holds the name of the class
* The middle part contains the attributes of the class
* The bottom part gives the methods or operations the class can take or undertake.



**Use case Diagram:**

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**Sequence diagram:**

A sequence diagram 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 object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**Collaboration diagram:**

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.



**Component Diagram:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

****

**Deployment Diagram:**

A **deployment diagram** in the Unified Modeling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.



**Activity Diagram:**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another

activity.The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent

uploadDataset()

preprocessDataset()

runSVM()

runCNN()

predictDisease()

**Data Flow Diagram:**

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

User

1. upload Dataset 2.upload dataset successfully

3. preprocess Dataset 4.Dataset preprocess successfully

5. Run SVM 6. Successfully Run SVM

7. Run CNN 8. Successfully Run CNN Algorithm

9. predict disease 10. Predicted disease successfully

**5. IMPLEMETATION**

**PYTHON**

**1.1 Introduction**

\* One of the most popular languages is Python. Guido van Rossum released this language in 1991. Python is available on the Mac, Windows, and Raspberry Pi operating systems. The syntax of Python is simple and identical to that of English. When compared to Python, it was seen that the other language requires a few extra lines.

\*It is an interpreter-based language because code may be run line by line after it has been written. This implies that rapid prototyping is possible across all platforms. Python is a big language with a free, binary-distributed interpreter standard library.

\* It is inferior to maintenance that is conducted and is straightforward to learn. It is an object-oriented, interpreted programming language. It supports several different programming paradigms in addition to object-oriented programming, including functional and procedural programming.

\* It supports several different programming paradigms in addition to object-oriented programming, including practical and procedural programming. Python is mighty while maintaining a relatively straightforward syntax. Classes, highly dynamic data types, modules, and exceptions are covered. Python can also be utilised by programmes that require programmable interfaces as an external language.

Here are some key features and characteristics of Python:

* Readability: Python emphasizes code readability with its clean and intuitive syntax. It uses indentation and whitespace to structure code blocks, making it easy to understand and maintain.
* Easy to Learn: Python's simplicity and readability make it an excellent choice for beginners. Its straightforward syntax and extensive documentation make it accessible for newcomers to programming.
* Interpreted Language: Python is an interpreted language, meaning that it doesn't need to be compiled before running. The Python interpreter reads and executes the code directly, making the development process faster and more interactive.
* Cross-platform Compatibility: Python is available for major operating systems like Windows, macOS, and Linux. This cross-platform compatibility allows developers to write code once and run it on different platforms without modifications.
* Large Standard Library: Python comes with a vast standard library that provides ready-to-use modules and functions for various tasks. It covers areas such as file I/O, networking, regular expressions, databases, and more, saving developers time and effort.
* Extensible and Modular: Python supports modular programming, enabling developers to organize code into reusable modules and packages. Additionally, Python allows integrating modules written in other languages, such as C or C++, providing flexibility and performance optimizations.
* Wide Range of Libraries and Frameworks: Python has a vibrant ecosystem with numerous third-party libraries and frameworks. These libraries, such as NumPy, pandas, TensorFlow, and Django, extend Python's capabilities for specific domains, making it a powerful tool for diverse applications.
* Object-Oriented: Python supports object-oriented programming (OOP) principles, allowing developers to create and work with classes and objects. OOP provides a structured approach to code organization, promoting code reuse and modularity.
* Dynamic Typing: Python is dynamically typed, meaning variable types are determined at runtime. Developers do not need to declare variable types explicitly, which enhances flexibility and simplifies code writing.

**1.2 Installation**

To install Python on your computer, follow these basic steps:

* Step 1: Visit the Python website Go to the official Python website at <https://www.python.org/>.
* Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
* Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
* Step 4: Install Python 3.7.0 using Cmd.
* Step 5: Install the all libraries that required to run the project
* Step 6: Run

**1.3 Python Features:**

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

**1.4 Python GUI (Tkinter)**

* Python provides a wide range of options for GUI development (Graphical User Interfaces).
* Tkinter, the most widely used GUI technique, is used for all of them.
* The Tk GUI toolkit offered by Python is used with the conventional Python interface.
* Tkinter is the easiest and quickest way to write Python GUI programs.
* Using Tkinter, creating a GUI is simple.
* A part of Python's built-in library is Tkinter. The GUI programs were created.
* Python and Tkinter together give a straightforward and quick way. The Tk GUI toolkit's object-oriented user interface is called Tkinter.

Making a GUI application is easy using Tkinter. Following are the steps:

1) Install the Tkinter module in place.

2) The GUI application Makes the primary window

3) Include one or more of the widgets mentioned above in the GUI application.

4) Set up the main event loop such that it reacts to each user-initiated event.

Although Tkinter is the only GUI framework included in the Python standard library, Python includes a GUI framework. The default library for Python is called Tkinter. Tk is a scripting language often used in designing, testing, and developing GUIs. Tk is a free, open-source widget toolkit that may be used to build GUI applications in a wide range of computer languages.

**1.5 Python IDLE**

* Python IDLE offers a full-fledged file editor, which gives you the ability to write and execute Python programs from within this program. The built-in file editor also includes several features, like code completion and automatic indentation, that will speed up your coding workflow.
* Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5. 2b1 release
* IDLE is used to execute statements similar to Python Shell. IDLE is used to create, modify, and execute Python code. IDLE provides a fully-featured text editor to write Python scripts and provides features like syntax highlighting, auto-completion, and smart indent.
* IDLE has two modes: interactive and script. We wrote our first program, “Hello, World!” in interactive mode. Interactive mode immediately returns the results of commands you enter into the shell. In script mode, you will write a script and then run it.
* The IDE Python IDLE is a good place to start as it helps you become familiar with the way Python works and understand its syntax. This IDE is good to start programming in Python due to its great debugger, but once you are fluent and start developing projects it is necessary to jump to another, more complete IDE.
* Python IDLE (Integrated Development and Learning Environment) is an interactive development environment included with the Python programming language. It provides a convenient way to write, execute, and debug Python code.

When you install Python, IDLE is typically installed along with it. To open IDLE, you can follow these steps:

* Open the command prompt (Windows) or terminal (macOS/Linux).
* Type "idle" and press Enter. Alternatively, you can specify the version with "idle3" or "idle2" for Python 3 or Python 2, respectively.
* Once IDLE is launched, you will see the Python shell, which is an interactive environment where you can type and execute Python code directly.

Here are some features and functionalities provided by Python IDLE:

* Editor: IDLE includes a text editor where you can write your Python code. It offers syntax highlighting, automatic indentation, and code completion to enhance your coding experience.
* Interactive Shell: The Python shell in IDLE allows you to execute Python code interactively. You can type commands, statements, or function calls directly in the shell, and Python will execute them immediately.
* Debugging: IDLE provides basic debugging capabilities to help you find and fix errors in your code. You can set breakpoints, step through code, inspect variables, and track the program's execution.
* Python Help: IDLE provides access to the Python documentation and built-in help. You can access the help menu to find information about Python modules, functions, classes, and more.
* Script Execution: In addition to the interactive shell, IDLE allows you to run Python scripts stored in files. You can write your code in the editor and execute it as a script to see the output or interact with the program.
* Customization: IDLE can be customized to suit your preferences. You can modify settings related to syntax highlighting, indentation, fonts, and more.
* Python IDLE serves as a beginner-friendly development environment and learning tool. It is suitable for writing small scripts, testing code snippets, experimenting with Python features, and learning the language's basics. However, for more advanced development projects, you may consider using other code editors or integrated development environments (IDEs) that provide additional features and better project management capabilities.

**1.6 Libraries**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

Python libraries are designed to solve common problems, such as handling data, performing mathematical operations, interacting with databases, working with files, implementing networking protocols, creating graphical user interfaces (GUIs), and much more. They provide ready-to-use functions, classes, and methods that simplify complex operations and save development time.

**Libraries in Python offer various advantages:**

* Code Reusability:
* Efficiency:
* Collaboration
* Domain-Specific Functionality
* To use a Python library, you need to install it first.

There are some libraries following:

* **Pandas:**

Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.

Pandas are a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. Pandas are a Python library used for working with data sets.

* It has functions for analysing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.
* Pandas allow us to analyse big data and make conclusions based on statistical theories.
* Pandas can clean messy data sets, and make them readable and relevant.

Relevant data is very important in data science. Pandas are a Python library for data analysis. Started by Wes McKinney in 2008 out of a need for a powerful and flexible quantitative analysis tool, pandas have grown into one of the most popular Python libraries. It has an extremely active community of contributors. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself.

* **NumPy:**

The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, Fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.
* In Python we have lists that serve the purpose of arrays, but they are slow to process.
* NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
* The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.
* Arrays are very frequently used in data science, where speed and resources are very important.
* **Matplotlib:**

It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization.

Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.

Here are some key features and functionalities of the Matplotlib library:

* Plotting Functions
* Customization Options
* Multiple Interfaces
* Integration with NumPy and pandas
* Subplots and Figures:
* Saving and Exporting
* **Scikit-learn:**

The most stable and practical machine learning library for Python is scikit-learn. Regression, dimensionality reduction, classification, and clustering are just a few of the helpful tools it provides through the Python interface for statistical modeling and machine learning. It is an essential part of the Python machine learning toolbox used by JP Morgan. It is frequently used in various machine learning applications, including classification and predictive analysis.

Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing.

Here are some key features and functionalities of the Scikit-learn library:

* Easy-to-Use Interface:
* Broad Range of Algorithms:
* Data Pre-processing and Feature Engineering:
* Model Evaluation and Validation:
* Integration with NumPy and pandas:
* Robust Documentation and Community Support:
* **Keras:**

\* Google's Keras is a cutting-edge deep learning API for creating neural networks. It is created in Python and is designed to simplify the development of neural networks. Additionally, it enables the use of various neural networks for computation. Deep learning models are developed and tested using the free and open-source Python software known as Keras.

Keras is a high-level deep learning library for Python. It is designed to provide a user-friendly and intuitive interface for building and training deep learning models. Keras acts as a front-end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.

Here are some key features and functionalities of the Keras library:

* User-Friendly API
* Multi-backend Support
* Wide Range of Neural Network Architectures
* Pre-trained Models and Transfer Learning:
* Easy Model Training and Evaluation:
* GPU Support:
* **h5py:**

\* The h5py Python module offers an interface for the binary HDF5 data format. Thanks to p5py, the top can quickly halt the vast amount of numerical data and alter it using the NumPy library. It employs common syntax for Python, NumPy, and dictionary arrays.

h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.

Here are some key features and functionalities of the h5py library:

* + HDF5 File Access
  + Dataset Handling:
  + Group Organization:
  + Attributes:
  + Compatibility with NumPy
  + Performance
* **Tensor flow**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.

TensorFlow is a popular open-source library for machine learning and deep learning. It provides a comprehensive set of tools, APIs, and computational resources for building and training various types of machine learning models, especially neural networks.

Here are some key features and functionalities of TensorFlow:

* Neural Network Framework:
* Computational Graphs
* Automatic Differentiation
* GPU and TPU Support
* Distributed Computing
* Deployment Capabilities
* **Tkinter**

Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Tkinter is a standard Python library used for creating graphical user interfaces (GUIs). It provides a set of modules and classes that allow you to develop interactive and visually appealing desktop applications.

Here are some key features and functionalities of Tkinter:

* Cross-Platform Compatibility
* Simple and Easy-to-Use
* Widgets and Layout Management
* Event-Driven Programming
* Customization and Styling
* Integration with Other Libraries
* **NLTK**

NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc NLTK (Natural Language Toolkit) is the go-to API for NLP (Natural Language Processing) with Python. It is a really powerful tool to pre-process text data for further analysis like with ML models for instance. It helps convert text into numbers, which the model can then easily work with.

NLTK (Natural Language Toolkit) is a Python library widely used for working with human language data and implementing natural language processing (NLP) tasks. It provides a set of tools, corpora, and resources for tasks such as tokenization, stemming, tagging, parsing, sentiment analysis, and more.

Here are some key features and functionalities of NLTK:

* Text Processing
* Part-of-Speech Tagging
* Named Entity Recognition
* Chunking and Parsing
* Sentiment Analysis:
* WordNet Integration:
* **Scipy**

SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

SciPy is a powerful scientific computing library for Python that provides a wide range of mathematical algorithms and functions. It builds upon NumPy, another fundamental library for numerical computing, and extends its capabilities by adding additional tools for scientific and technical computing tasks.

Here are some key features and functionalities of SciPy:

* Numerical Integration:
* Optimization and Root Finding
* Linear Algebra
* Signal and Image Processing
* Statistics

**5.2 Sample Code:**

import tkinter

from tkinter import \*

from tkinter import ttk

import matplotlib.pyplot as plt

import numpy as np

from PIL import Image,ImageTk

from tkinter import filedialog

from sklearn import svm

import numpy as np

from tkinter.filedialog import askopenfilename

import os

import cv2

from sklearn.metrics import f1\_score

import seaborn as sns

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

import pandas as pd

from keras.utils.np\_utils import to\_categorical

from keras.models import Sequential, load\_model

import pickle

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from keras.callbacks import ModelCheckpoint

from keras.layers import MaxPooling2D

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.layers import Convolution2D

from keras.layers import Dense, Dropout, Activation, Flatten

global filename, canvas, text, images, root

global X, Y, filename, X\_train, X\_test, y\_train, y\_test, cnn\_model

labels = ['vein1', 'vein2', 'vein3', 'vein4', 'vein5', 'vein6', 'vein7', 'vein8', 'vein9', 'vein10', 'vein11',

'vein12', 'vein13', 'vein14', 'vein15', 'vein16', 'vein17', 'vein18', 'vein19', 'vein20',

'vein21', 'vein22', 'vein23', 'vein24', 'vein25', 'vein26', 'vein27', 'vein28', 'vein29', 'vein30']

global accuracy, precision, recall, fscore

def getID(name):

index = 0

for i in range(len(labels)):

if labels[i] == name:

index = i

break

return index

def uploadDataset():

global filename

filename = filedialog.askdirectory(initialdir=".")

text.delete('1.0', END)

text.insert(END,filename+" dataset loaded\n\n")

Y = np.load('model/Y.txt.npy')

label = ['P1','P2','P3','P4','P5','P6','P7','P8','P9','P10','P11','P12','P13','P14','P15','P16','P17','P18','P19','P20','P21',

'P22','P23','P24','P25','P26','P27','P28','P29','P30']

unique, count = np.unique(Y, return\_counts = True)

height = count

bars = label

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.xlabel("Person ID")

plt.ylabel("Images Count")

plt.title("Different Person Finger Vein found in Dataset")

plt.show()

def preprocessDataset():

global X, Y, filename, X\_train, X\_test, y\_train, y\_test

text.delete('1.0', END)

if os.path.exists("model/X.txt.npy"):

X = np.load('model/X.txt.npy')

Y = np.load('model/Y.txt.npy')

else:

X = []

Y = []

for root, dirs, directory in os.walk(filename):

for j in range(len(directory)):

name = os.path.basename(root)

if 'Thumbs.db' not in directory[j]:

img = cv2.imread(root+"/"+directory[j])

img = cv2.resize(img, (32, 32))

im2arr = np.array(img)

im2arr = im2arr.reshape(32, 32, 3)

X.append(im2arr)

label = getID(name)

Y.append(label)

X = np.asarray(X)

Y = np.asarray(Y)

np.save('model/X.txt',X)

np.save('model/Y.txt',Y)

X = X.astype('float32')

X = X/255

indices = np.arange(X.shape[0])

np.random.shuffle(indices)

X = X[indices]

Y = Y[indices]

Y = to\_categorical(Y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2) #split dataset into train and test

text.insert(END,"Total Images found in dataset : "+str(X.shape[0])+"\n\n")

text.insert(END,"Dataset Train & Test Split Details\n\n")

text.insert(END,"80% dataset used for training : "+str(X\_train.shape[0])+"\n")

text.insert(END,"20% dataset used for training : "+str(X\_test.shape[0])+"\n")

text.update\_idletasks()

test = X[3]

cv2.imshow("Sample Processed Image",cv2.resize(test,(150,250)))

cv2.waitKey(0)

#function to calculate all metrics

def calculateMetrics(algorithm, testY, predict):

p = precision\_score(testY, predict,average='macro') \* 100

r = recall\_score(testY, predict,average='macro') \* 100

f = f1\_score(testY, predict,average='macro') \* 100

a = accuracy\_score(testY,predict)\*100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

text.insert(END,algorithm+" Accuracy : "+str(a)+"\n")

text.insert(END,algorithm+" Precision : "+str(p)+"\n")

text.insert(END,algorithm+" Recall : "+str(r)+"\n")

text.insert(END,algorithm+" FSCORE : "+str(f)+"\n\n")

conf\_matrix = confusion\_matrix(testY, predict)

labels = ['P1','P2','P3','P4','P5','P6','P7','P8','P9','P10','P11','P12','P13','P14','P15','P16','P17','P18','P19','P20','P21',

'P22','P23','P24','P25','P26','P27','P28','P29','P30']

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

def runSVM():

global X\_train, X\_test, y\_train, y\_test

global accuracy, precision, recall, fscore

text.delete('1.0', END)

accuracy = []

precision = []

recall = []

fscore = []

X\_train1 = np.reshape(X\_train, (X\_train.shape[0], (X\_train.shape[1] \* X\_train.shape[2] \* X\_train.shape[3])))

X\_test1 = np.reshape(X\_test, (X\_test.shape[0], (X\_test.shape[1] \* X\_test.shape[2] \* X\_test.shape[3])))

y\_train1 = np.argmax(y\_train, axis=1)

y\_test1 = np.argmax(y\_test, axis=1)

X\_train1 = X\_train1[0:300,0:100]

X\_test1 = X\_test1[0:100,0:100]

y\_train1 = y\_train1[0:300]

y\_test1 = y\_test1[0:100]

svm\_cls = svm.SVC()

svm\_cls.fit(X\_train1, y\_train1)

predict = svm\_cls.predict(X\_test1)

calculateMetrics("SVM Algorithm", y\_test1, predict)

def runCNN():

global X\_train, X\_test, y\_train, y\_test

global accuracy, precision, recall, fscore, cnn\_model

cnn\_model = Sequential()

cnn\_model.add(Convolution2D(32, (3, 3), input\_shape = (X\_train.shape[1], X\_train.shape[2], X\_train.shape[3]), activation = 'relu'))

cnn\_model.add(MaxPooling2D(pool\_size = (2, 2)))

cnn\_model.add(Convolution2D(32, (2, 2), activation = 'relu'))

cnn\_model.add(MaxPooling2D(pool\_size = (3, 3)))

cnn\_model.add(Flatten())

cnn\_model.add(Dense(units = 256, activation = 'relu'))

cnn\_model.add(Dense(units = y\_train.shape[1], activation = 'softmax'))

cnn\_model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

if os.path.exists("model/model\_weights.hdf5") == False:

model\_check\_point = ModelCheckpoint(filepath='model/model\_weights.hdf5', verbose = 1, save\_best\_only = True)

hist = cnn\_model.fit(X\_train, y\_train, batch\_size = 16, epochs = 80, validation\_data=(X\_test, y\_test), callbacks=[model\_check\_point], verbose=1)

else:

cnn\_model = load\_model("model/model\_weights.hdf5")

predict = cnn\_model.predict(X\_test)

predict = np.argmax(predict, axis=1)

y\_test1 = np.argmax(y\_test, axis=1)

calculateMetrics("CNN Algorithm", y\_test1, predict)

def graph():

df = pd.DataFrame([['SVM','Precision',precision[0]],['SVM','Recall',recall[0]],['SVM','F1 Score',fscore[0]],['SVM','Accuracy',accuracy[0]],

['CNN','Precision',precision[1]],['CNN','Recall',recall[1]],['CNN','F1 Score',fscore[1]],['CNN','Accuracy',accuracy[1]],

],columns=['Parameters','Algorithms','Value'])

df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')

plt.show()

def predictDisease(filename):

global cnn\_model

text.delete('1.0', END)

image = cv2.imread(filename)

img = cv2.resize(image, (80, 80))

im2arr = np.array(img)

im2arr = im2arr.reshape(1,80,80,3)

img = np.asarray(im2arr)

img = img.astype('float32')

img = img/255

preds = cnn\_model.predict(img)

predict = np.argmax(preds)

max\_value = np.amax(preds)

print(max\_value)

if max\_value > 0.60:

img = cv2.imread(filename)

img = cv2.resize(img, (700,300))

cv2.putText(img, 'Person Vein Identified as : '+labels[predict], (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.9, (0, 0, 255), 2)

text.insert(END,'Person Vein Identified as : '+labels[predict]+"\n")

else:

img = cv2.imread(filename)

img = cv2.resize(img, (700,400))

cv2.putText(img, 'Unable to identify person vein', (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.9, (0, 0, 255), 2)

text.insert(END,'Unable to identify person vein : '+labels[predict])

cv2.imwrite("images/output.png", img)

def predict():

global canvas, images, root

filename = filedialog.askopenfilename(initialdir="testImages")

predictDisease(filename)

img = Image.open("images/output.png")

img = img.resize((700, 300))

picture = ImageTk.PhotoImage(img)

canvas.configure(image = picture)

canvas.image = picture

root.update\_idletasks()

def Main():

global text, canvas, images, root

root = tkinter.Tk()

root.geometry("1300x1200")

root.title("Person Vein Identification using CNN")

root.resizable(True,True)

font = ('times', 14, 'bold')

title = Label(root, text='Person Vein Identification using CNN')

title.config(bg='yellow3', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 12, 'bold')

img = Image.open("images/download.png")

img.resize((600, 300))

picture = ImageTk.PhotoImage(img)

canvas = Label(root, image=picture)

canvas.place(x=300,y=200)

uploadButton = Button(root, text="Upload Finger Vein Dataset", command=uploadDataset)

uploadButton.place(x=60,y=80)

uploadButton.config(font=font1)

preprocessButton = Button(root, text="Preprocess Dataset", command=preprocessDataset)

preprocessButton.place(x=400,y=80)

preprocessButton.config(font=font1)

svmButton = Button(root, text="Run SVM Algorithm", command=runSVM)

svmButton.place(x=600,y=80)

svmButton.config(font=font1)

cnnButton = Button(root, text="Run CNN Algorithm", command=runCNN)

cnnButton.place(x=60,y=130)

cnnButton.config(font=font1)

graphButton = Button(root, text="Comparison Graph", command=graph)

graphButton.place(x=400,y=130)

graphButton.config(font=font1)

predictButton = Button(root, text="Identify Finger Vein from Test Image", command=predict)

predictButton.place(x=600,y=130)

predictButton.config(font=font1)

text=Text(root,height=10,width=140)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=510)

root.mainloop()

if \_\_name\_\_== '\_\_main\_\_' :

Main ()

**6. TESTING**

**Implementation and Testing:**

Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

## **Implementation**

## The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

## **Testing**

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period.

### **System Testing**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

**Module Testing**

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

**Integration Testing**

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

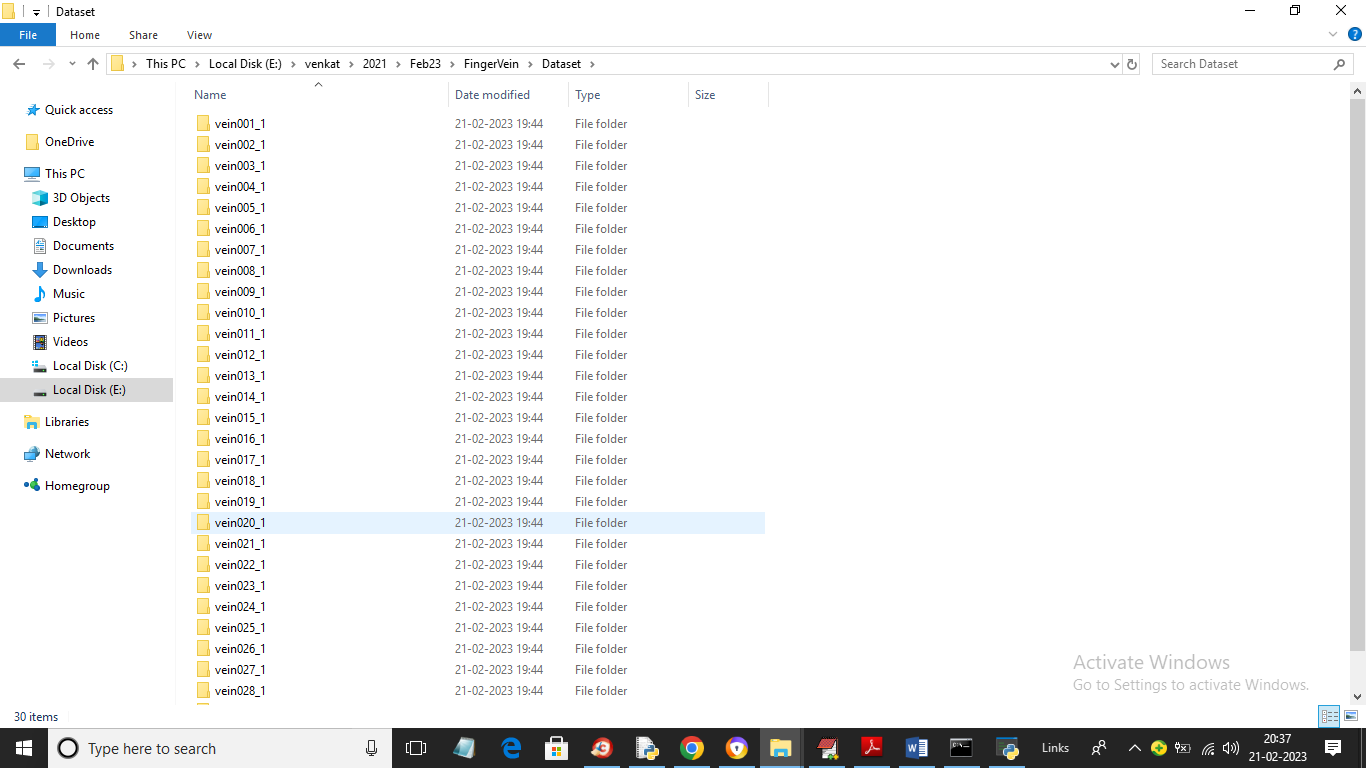
**Acceptance Testing**

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation

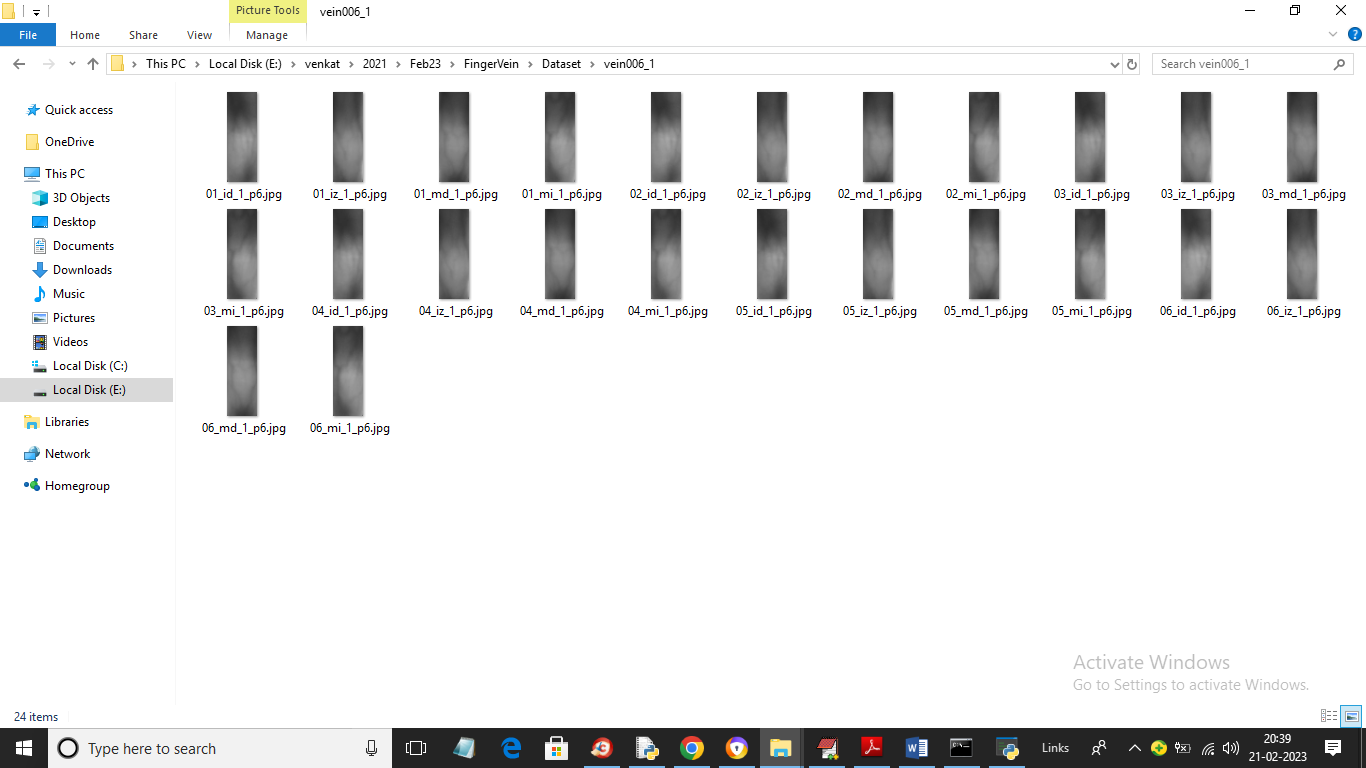
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload Dataset | Test Dataset is uploaded or not into the system | If the Dataset may not uploaded | We cannot do further operations | Datatset uploaded we will do further operations | High | High |
| 02 | Preprocess& Normalized Dataset | Test whether the Pre-process & NormalizedDataset Successfully or not | If the Pre-process & Normalized Dataset may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 03 | Feature Selection | Test whether the Feature Selection  Successfully or not | If the  Feature Selection  may not Successfully | We cannot do further operations | we will do further  operations | High | High |
| 04 | Run SVM Algorithm | Test wheather SVM Algorithm Run Successfully or not | If the  SVM Algorithm may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 04 | Run CNN Algorithm | Test wheather CNN Algorithm Run Successfully or not | If the  CNN Algorithm may not Run Successfully | We cannot do further operations | we will do further  operations | High | High |
| 05 | Graph | Test wheather Graph drawn Successfully or not | If the Graph may not Run Successfully | We cannot do further operations | We will do further operations | High | High |
| 05 | Predict Disease | Test Disese Prediction whether Successfully or not | If the  Predict of disease may not Successfully | We cannot do further operations | we will do further  operations | High | High |

**7.SCREENSHOTS**:

In below screen we are showing dataset details used in this project

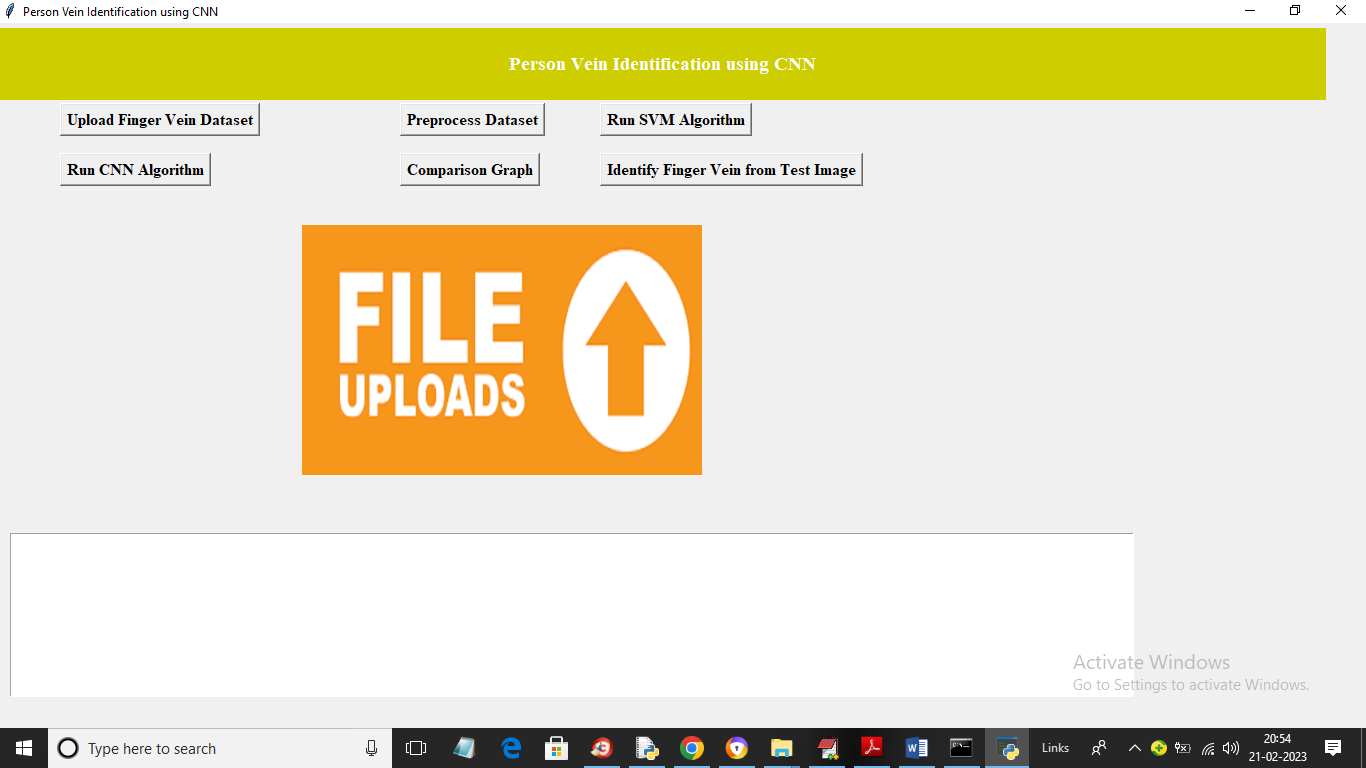


In above dataset screen we have 30 folders from 1 to 30 for persons finger vein from 1 to 30 and just go inside any folder to view finger vein images

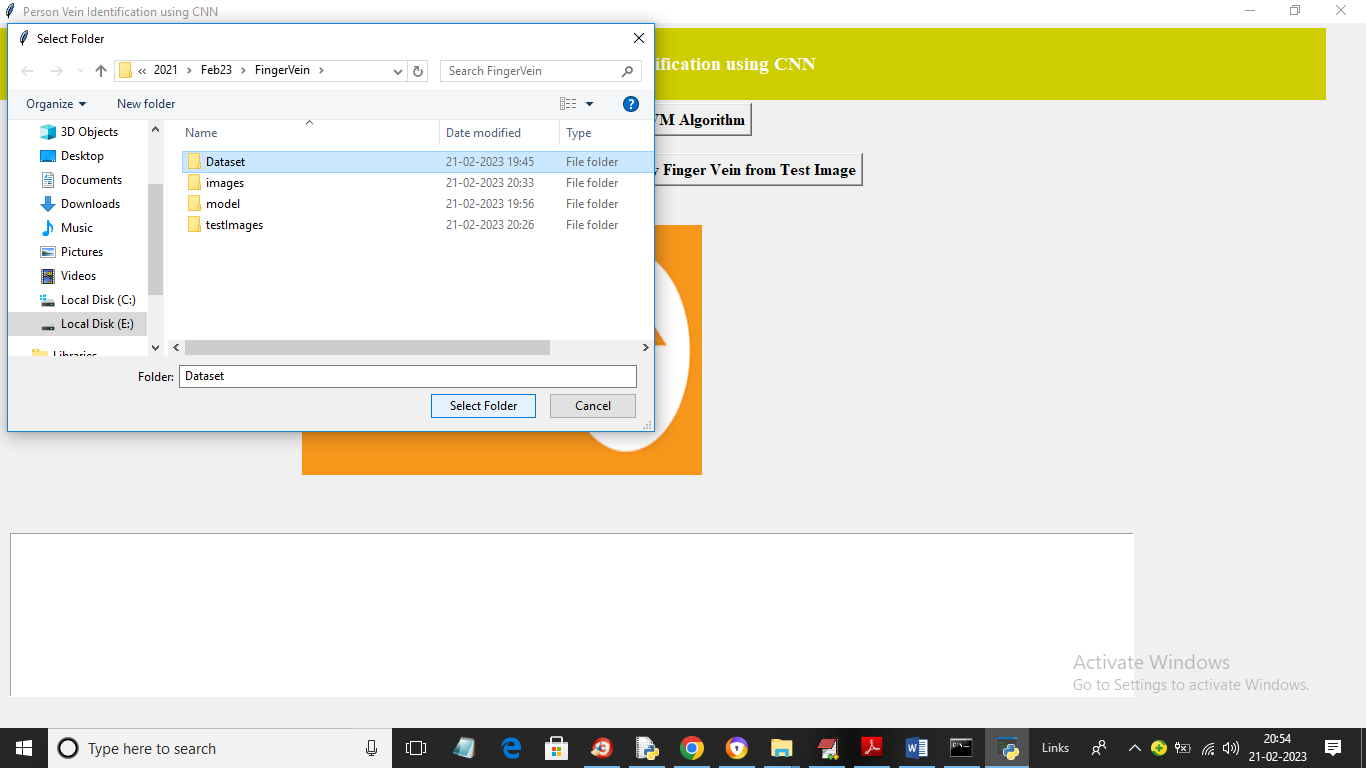


In above screen we can see images of finger vein and by using above images we will trained both SVM and CNN algorithms.

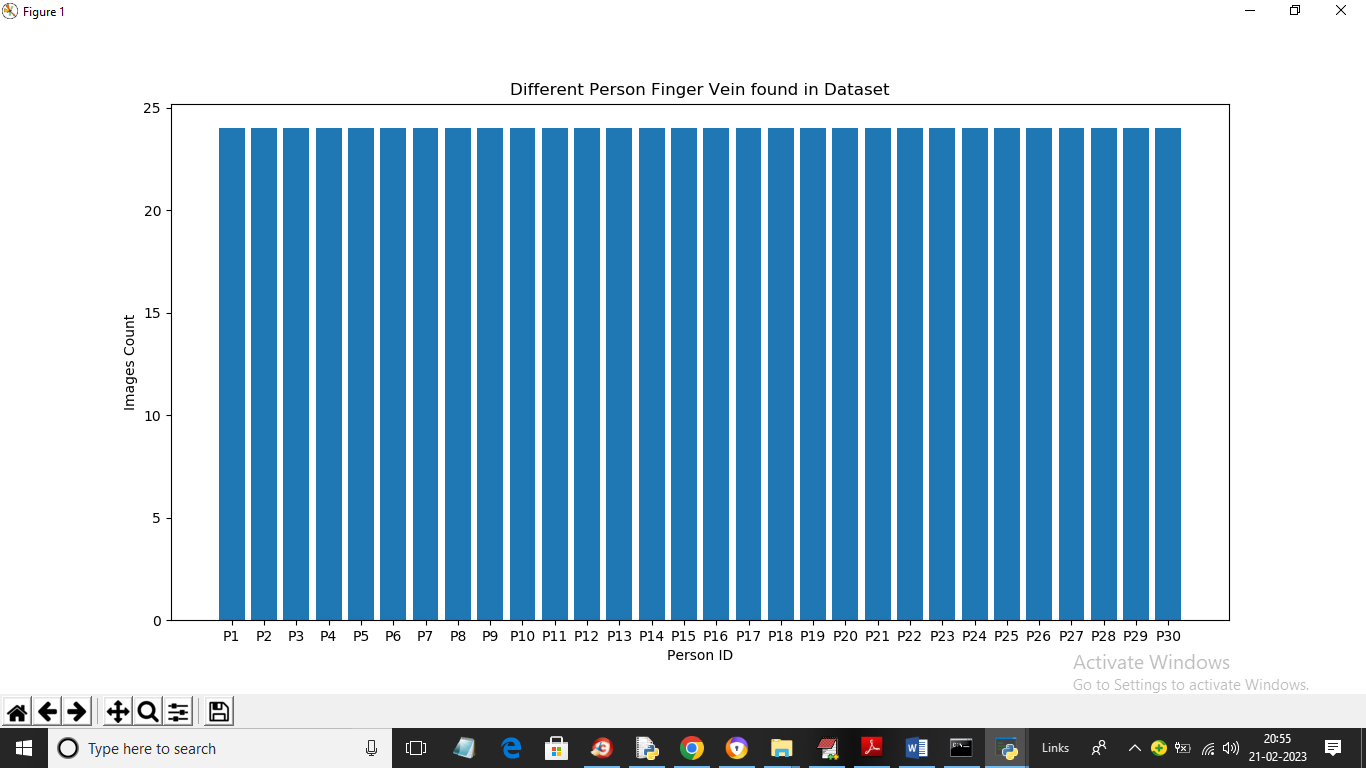
To run project double click on ‘run.bat’ file to get below page



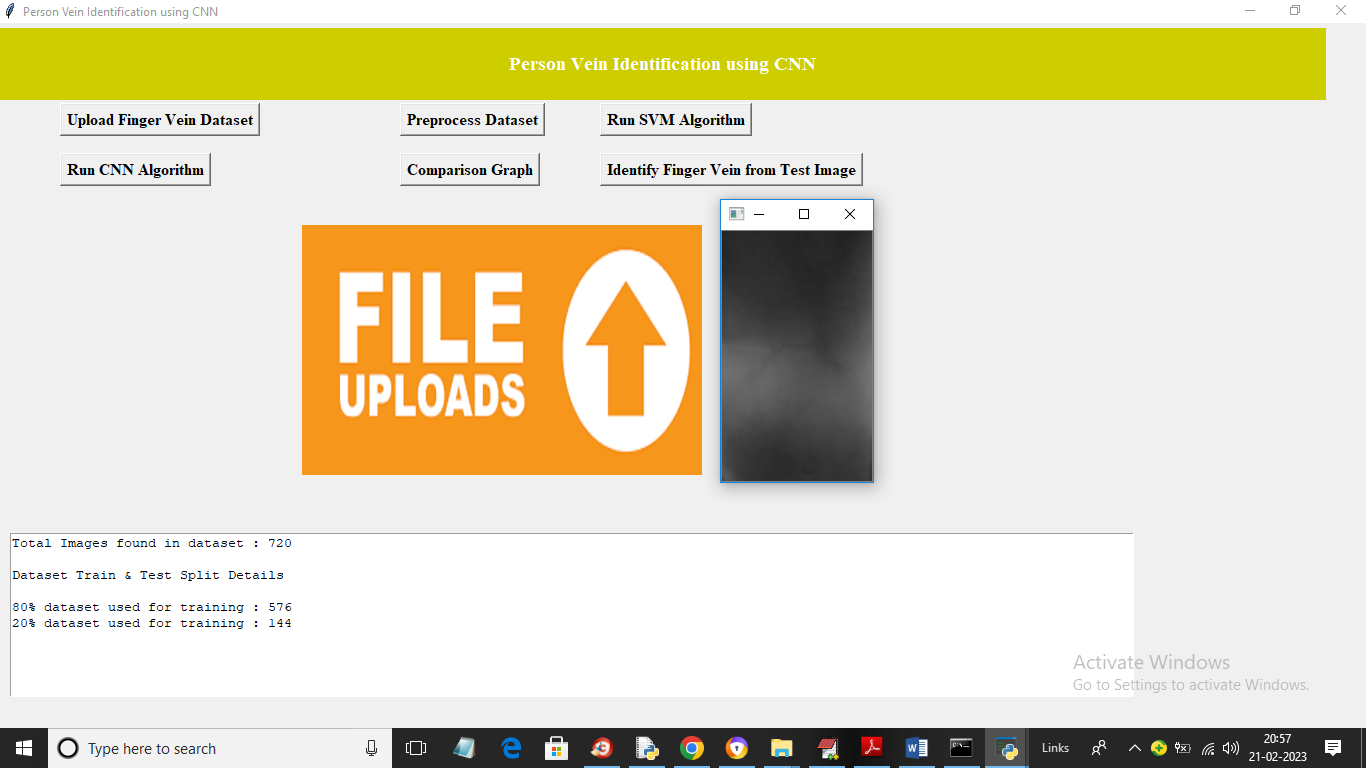
In above screen click on ‘Upload Finger Vein Dataset’ button to upload dataset and get below output



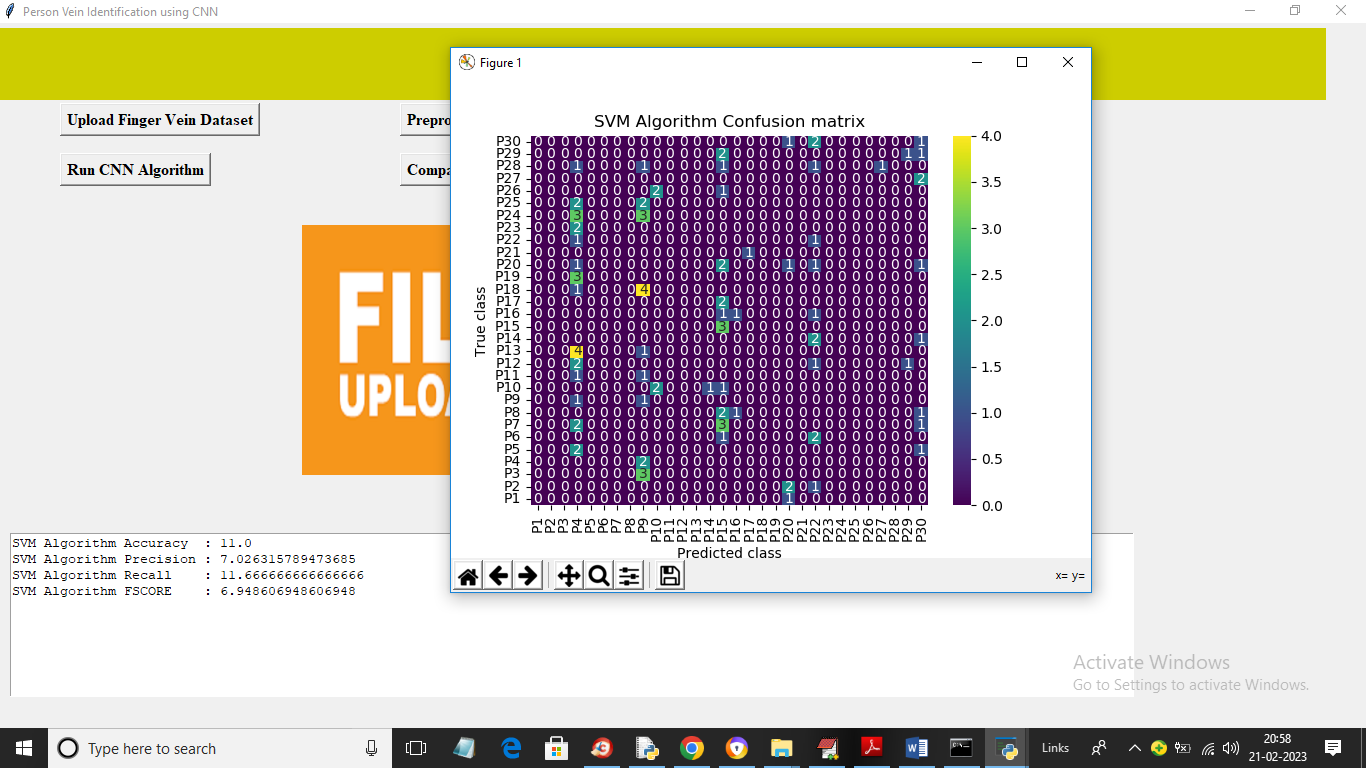
In above screen selecting and uploading ‘Dataset’ folder and then click on ‘Select Folder’ button to load dataset and get below output



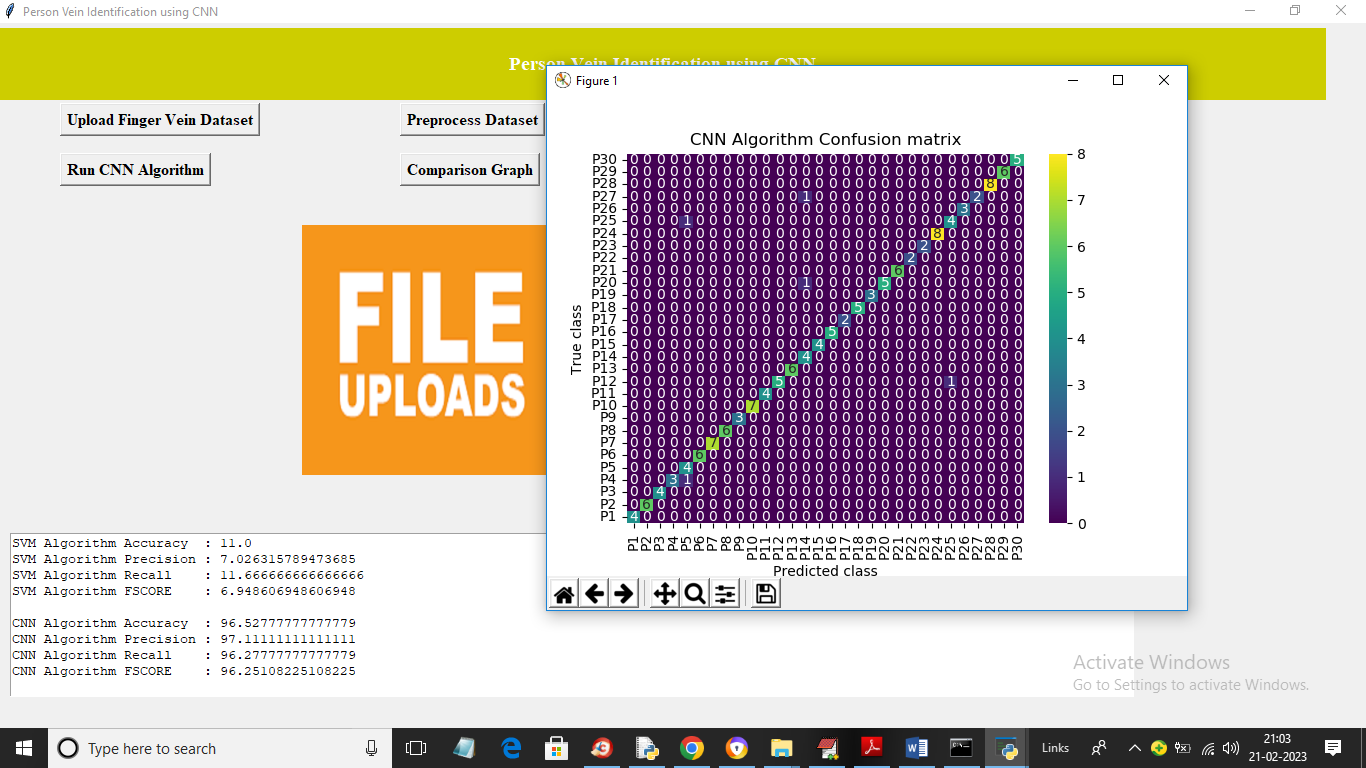
In above screen dataset loaded and in above graph x-axis represents person ID and y-axis represents number of finger images found for that person and now close above graph and then click on ‘Preprocess Dataset’ button to process image and get below output



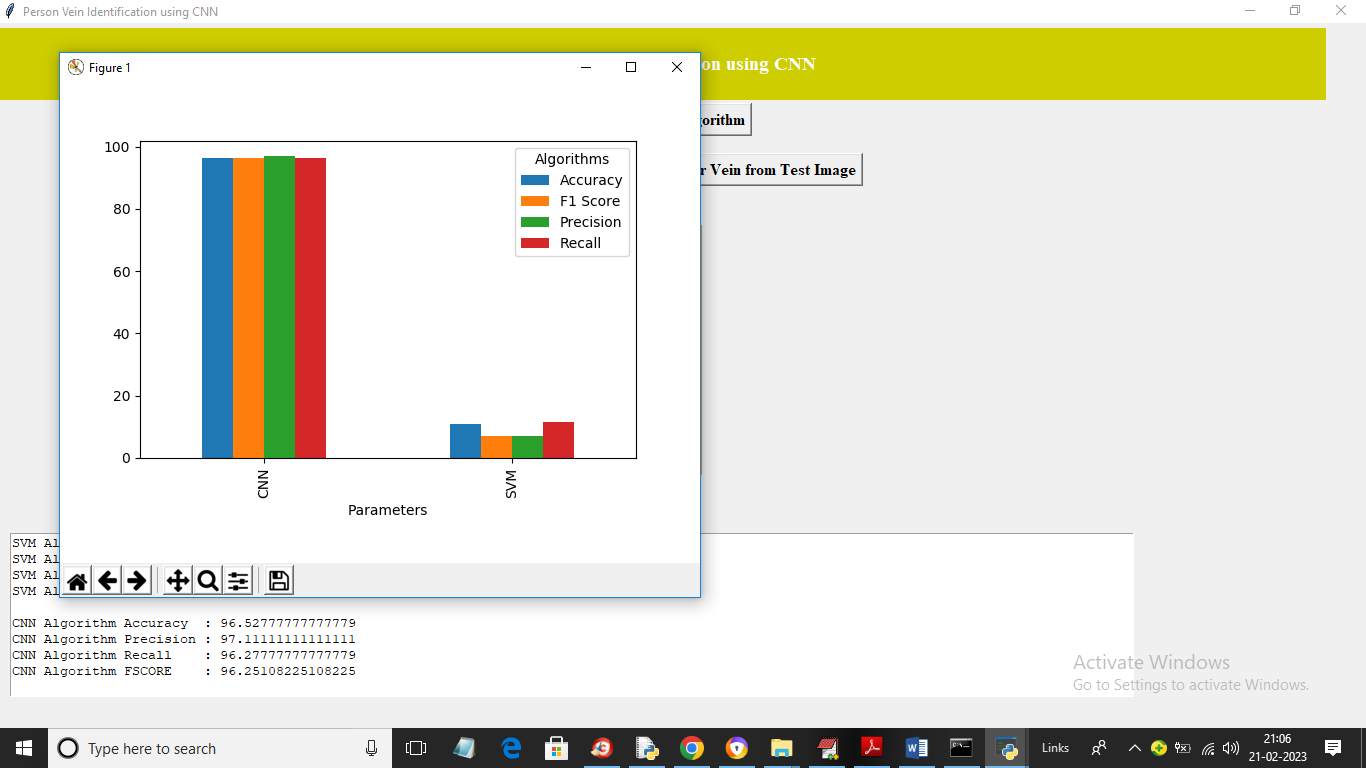
In above screen in text area we can see total processed images and can see 80% images size using for training and 20% for testing and then showing one processed image and now close above image and then click on ‘Run SVM Algorithm’ button to train SVM and get below output



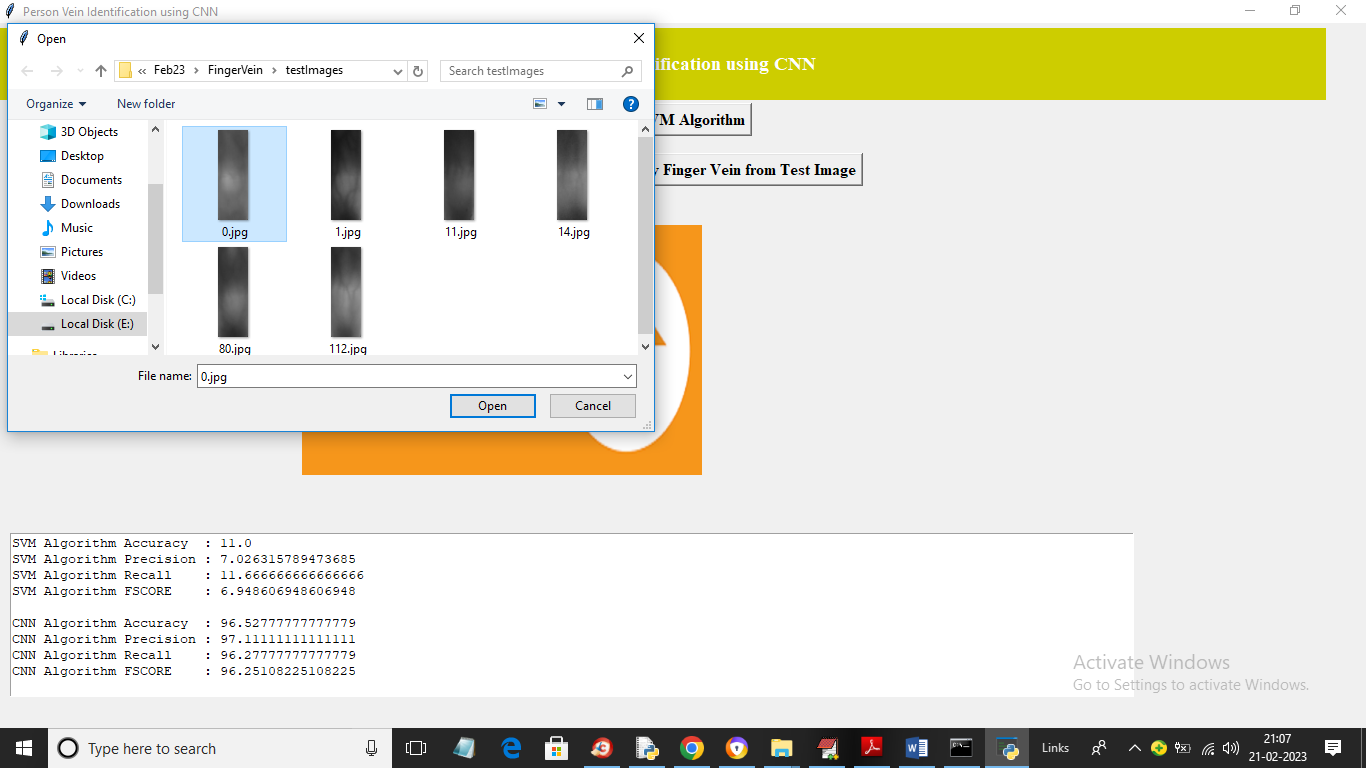
In above screen with SVM we got 11% accuracy and in confusion matrix graph x-axis represents Predicted Labels and y-axis represents True Labels and only the count in diagnol are the correct prediction count but in above graph diagnol we are seeing very few count and remaining are 0 so SVM is not accurate and now close above graph and then click on “Run CNN Algorithm’ button to train CNN and get below output



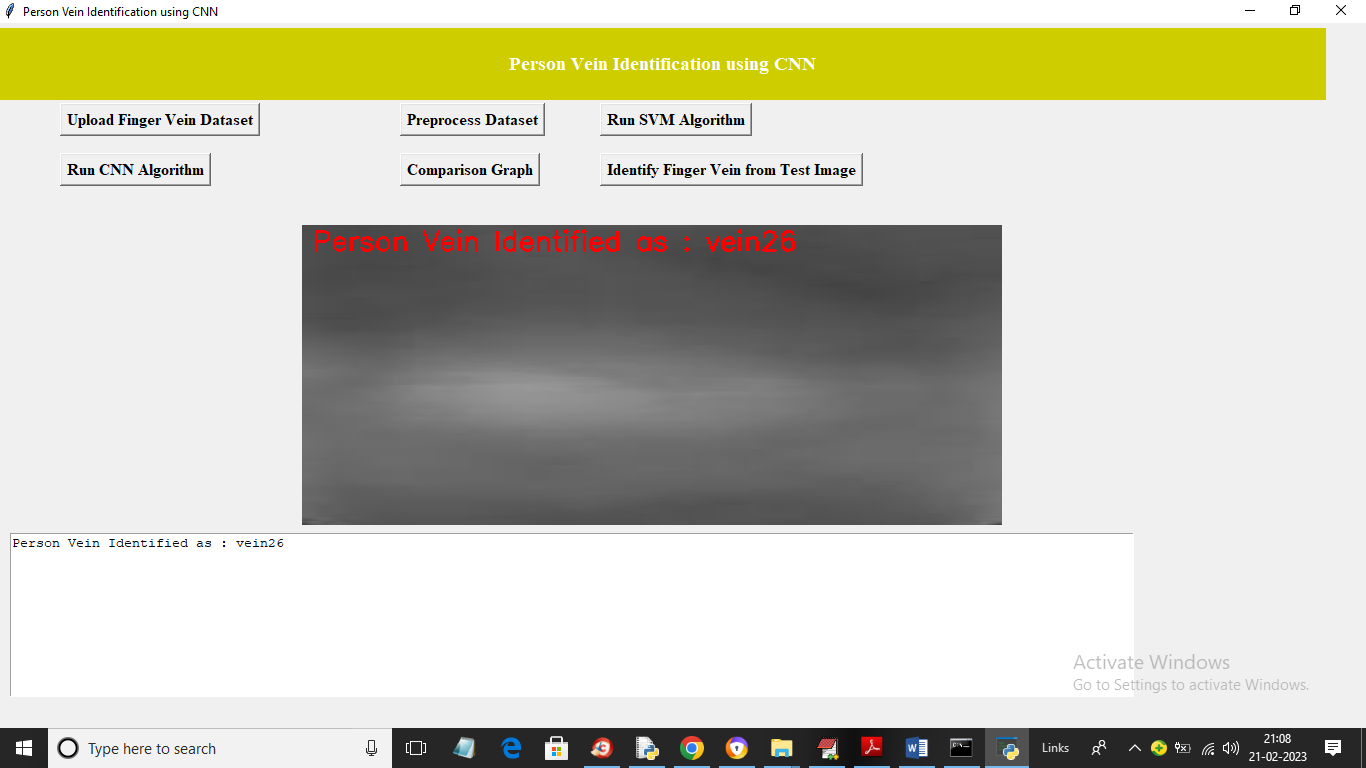
In above screen with CNN we got 98% accuracy and in confusion matrix graph in diagnol we can see all person images are correctly predicted and remaining blue colour boxes as in-corrected prediction contains 0 only. All blue colour boxes represents incorrect prediction count. Now close above graph and then click on ‘Comparison Graph’ button to get below graph



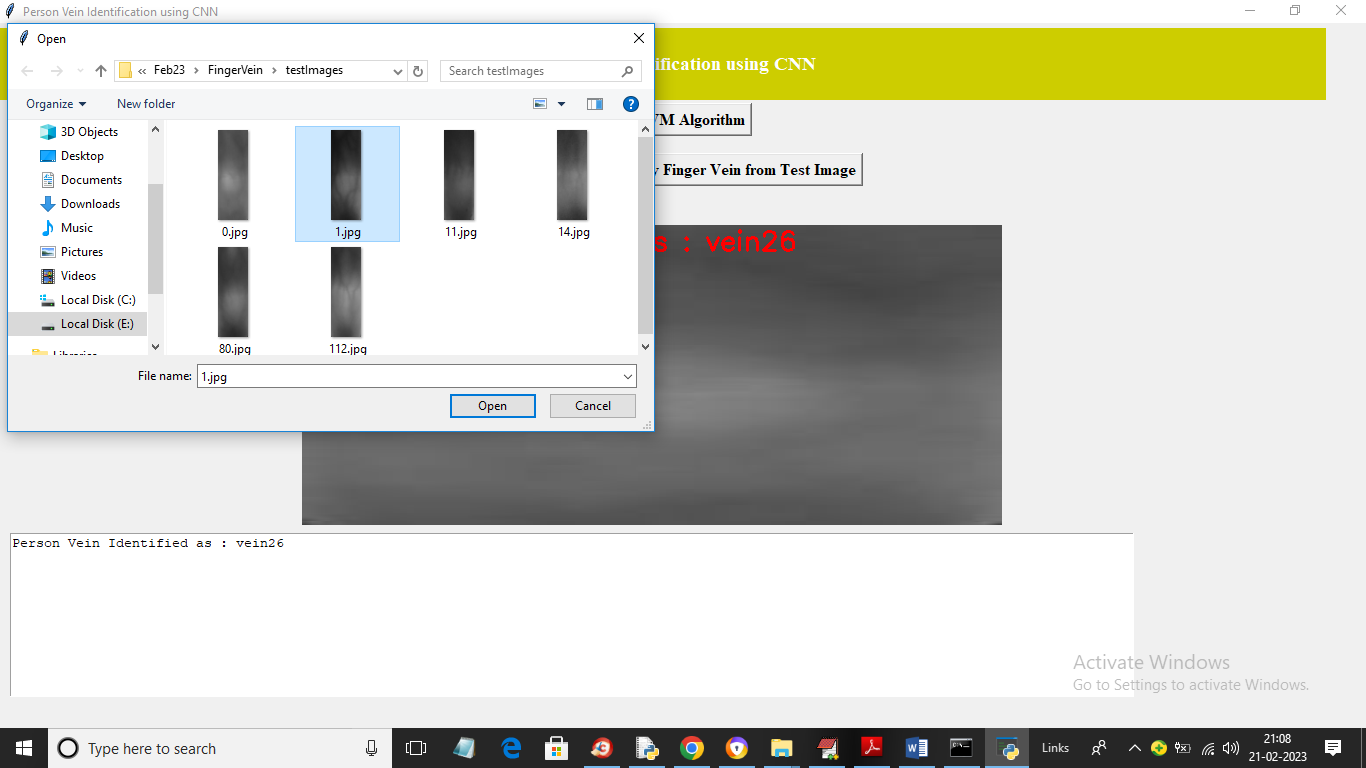
In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in both algorithms we can see CNN got high performance. Now close above graph and then click on ‘Identify Finger Vein from Test Image’ button to upload test finger vein image and then identify person



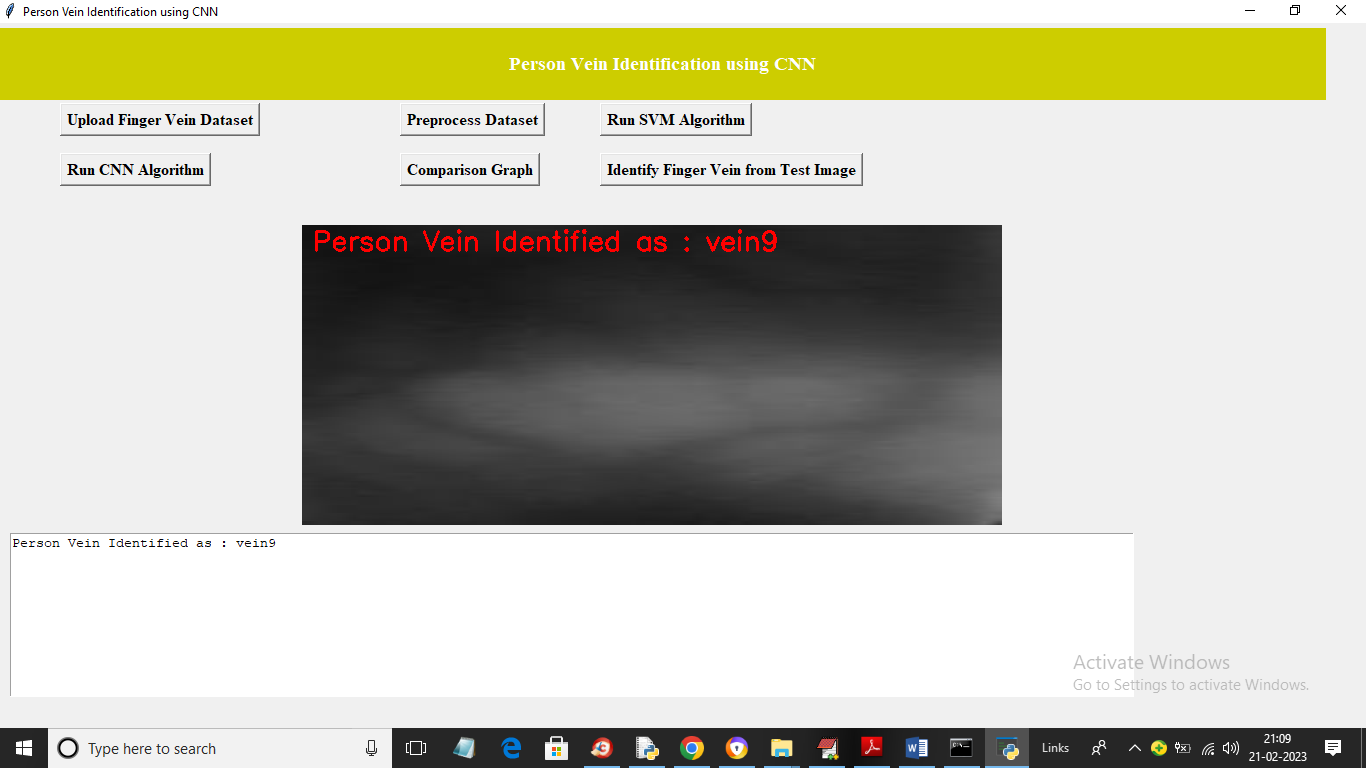
In above screen selecting and uploading ‘0.jpg’ and then click on ‘Open’ button to get below output

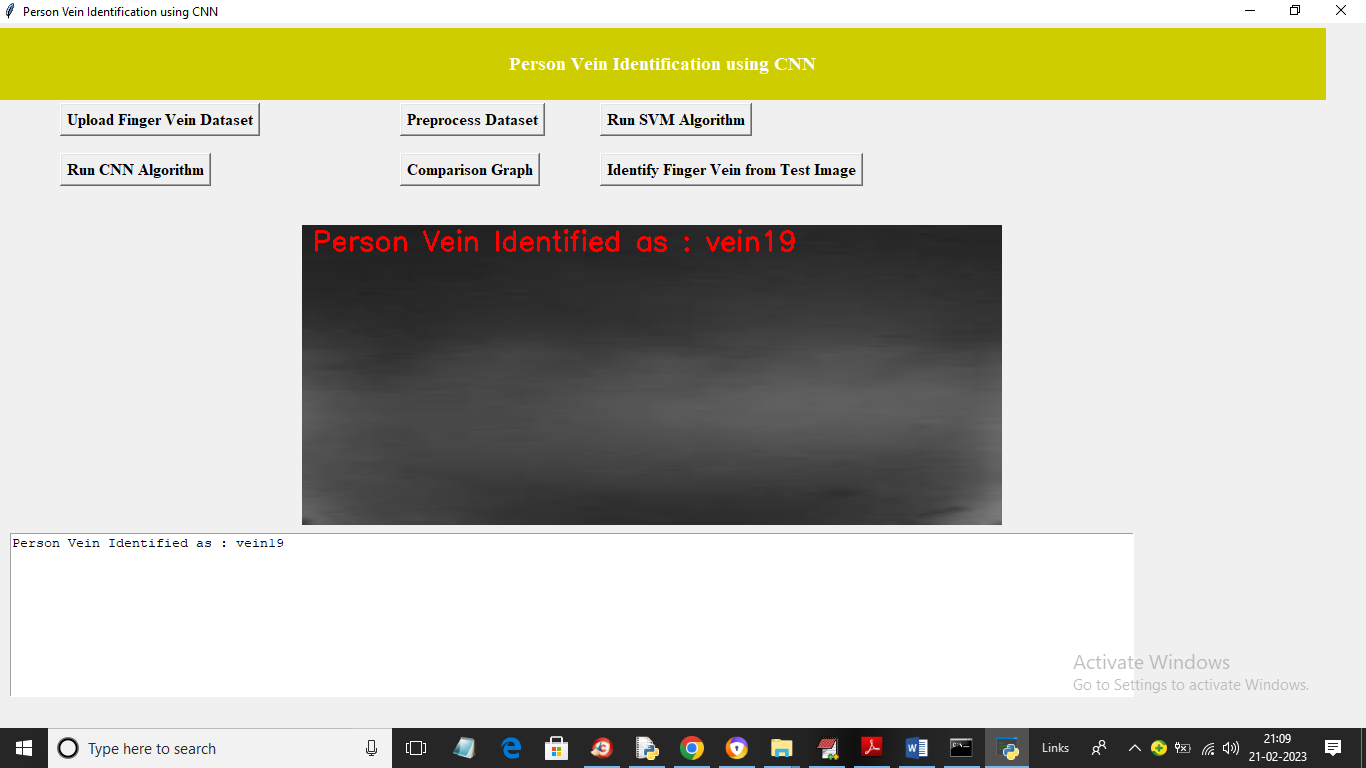


In above screen finger vein identified for person ID 26 and similarly you can upload and test other images



In above screen uploading another image and below is the output





**8. CONCLUSION:**

The primary goal of the suggested model is to raise the recognition rate by employing a small CNN to train the entire network. In this study, we provided a novel method based on the deep learning model to achieve personal identification through finger vein patterns. The uniqueness of the training data contributes to CNN's supremacy. Simple vein structures and modest databases play a role in the training process for finger vein detection. Recently, there has been consideration of an architecture called Merged CNN, which combines many networks to do different tasks. Instead of employing a single image in a multi-network model for this study, we combined the outputs of many finger vein images that were created using the same architecture. When creating a CNN model, there are no hard and fast guidelines. We routinely looked at opportunities and made decisions that appear absolutist, such the activation function, filter size, and layer count. We have examined our architecture's generalization performance and have seen notable advancements. Three open databases were used to evaluate the developed model. The Merged CNN provides supplementary information in comparison to a single CNN, which helps in identification that is more accurate. Our approach is successful and competitive. It's possible that different architectures with the capacity to enhance recognition for a limited set of training photos (less than three) may be employed in the future. However, a novel method for improving the quality of finger vein images can be used to increase personal identification.

**9. REFERENCES:**

[1] N. Miura, A. Nagasaka, and T. Miyatake, “Extraction of finger-vein patterns using maximum curvature points in image profiles,” Proc. IAPR Conf. Machine Vis. & Appl. pp. 347-350, Tsukuba Science City, May 2005.

[2] X. Wu, R. He, Z. Sun and T. Tan, “A light CNN for deep face representation with noisy labels,” arXiv:1151.02683v2, 2016.

[3] A. Kumar and Y. Zhou, “Human identification using finger images,” IEEE Trans. Image Process., vol. 21, pp. 2228-2244, Apr. 2012

[4] The Hong Kong Polytechnic University Finger Image Database (Version 1.0), http://www.comp.polyu.edu.hk/~csajaykr/fvdatavase.htm, 2012

[5] D. M. Weber and D. Casasent, “Quadratic Gabor filters for object detection,” IEEE Trans. Image Process., vol. 10, pp. 2180230, Feb. 2001

[6] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. arXiv preprint arXiv:1503.03832, 2015.

[7] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014

[8] D. Chen, X. Cao, L. Wang, F. Wen and J. Sun, “Bayesian face revisited: A joint formulation,” Proc. ECCV 2012, 2012.

[9] N. Miura, A. Nagasaka, and T. Miyatake, “Feature extraction of fingervein patterns based on repeated line tracking and its application to personal identification,” Machine Vis. & Appl., pp. 194-203, Jul, 2004

[10] Z. Zhao and A. Kumar, “Accurate periocular recognition under less constrained environment using semantics-assisted convolutional neural network,” IEEE Trans. Info. Forensics & Security, May, 2017.

[11] L. Dong, G. Yang, Y. Yin, F. Liu and X. Xi. “Finger vein verification based on a personalized best patches map,” Proc. 2nd IJCB 2014, Tampa, Sep.-Oct. 2014.

[12] L. Chen, J. Wang, S. Yang and H. He, “A finger vein image-based personal identification system with self-adaptive illuminance control,” IEEE Trans. Instrumentation & Measurement, 2017.

[13] Y. Lu, S. Yoon and D. S. Park, “Finger vein identification system using two cameras,” Electronics Letters, 2014.

[14] C. Liu and Y. H. Kim, “An efficient finger-vein extraction algorithm based on random forest regression with efficient local binary patterns,” Proc. ICIP 2016, 2016.