**DEEP HASHING FOR SECURE MULTIMODAL BIOMETRICS**

**ABSTRACT:**

When compared to unimodal systems, multimodal biometric systems have several advantages, including lower error rate, higher accuracy, and larger population coverage. However, multimodal systems have an increased demand for integrity and privacy because they must store multiple biometric traits associated with each user. In this paper, we present a deep learning framework for feature-level fusion that generates a secure multimodal template from each user's face and iris biometrics. We integrate a deep hashing (binarization) technique into the fusion architecture to generate a robust binary multimodal shared latent representation. Further, we employ a hybrid secure architecture by combining cancelable biometrics with secure sketch techniques and integrate it with a deep hashing framework, which makes it computationally prohibitive to forge a combination of multiple biometrics that passes the authentication. The efficacy of the proposed approach is shown using a multimodal database of face and iris and it is observed that the matching performance is improved due to the fusion of multiple biometrics. Furthermore, the proposed approach also provides cancelability and unlinkability of the templates along with improved privacy of the biometric data. Additionally, we also test the proposed hashing function for an image retrieval application using a benchmark dataset. The main goal of this paper is to develop a method for integrating multimodal fusion, deep hashing, and biometric security, with an emphasis on structural data from modalities like face and iris. The proposed approach is in no way a general biometrics security framework that can be applied to all biometrics modalities, as further research is needed to extend the proposed framework to other unconstrained biometric modalities.

**INTRODUCTION**

BIOMETRICS are difficult to forge, and unlike in traditional password-based access control systems, they do not have to be remembered. As much as these characteristics provide an advantage, they also create challenges related to protecting biometrics in the event of identity theft or a database compromise as each biometric characteristic is distinct and cannot be replaced by a newly generated arbitrary biometric. There are serious concerns about the security and privacy of an individual because of the proliferation of biometric usage. These concerns cannot be alleviated by using conventional cryptographic hashing as in case of alpha-numeric passwords.

Manuscript received September 5, 2018; revised April 5, 2019, May 3, 2020, and September 18, 2020; accepted September 30, 2020. Date of publication October 22, 2020; date of current version November 9, 2020. This material is based upon work supported by the Center for Identification Technology Research (CITeR) and the National Science Foundation (NSF) under Grant No. 1650474. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Julien Bringer. (Corresponding author: Veeru Talreja.)

because the cryptographic hashes are extremely sensitive to noise and are not suitable for the protection of biometrics due to inherent variability and noise in biometric measurements.

The leakage of biometric information to an adversary constitutes a serious threat to security and privacy because if an adversary gains access to a biometric database, he can potentially obtain the stored user information. The attacker can use this information to gain unauthorized access to the system by reverse engineering the system and creating a physical spoof. Furthermore, an attacker can abuse the biometric information for unintended purposes and violate user privacy

Multimodal biometric systems use a combination of different biometric traits such as face and iris, or face and fingerprint. Multimodal systems are generally more resistant to spoofing attacks [2]. Moreover, multimodal systems can be made to be more universal than unimodal systems, since the use of multiple modalities can compensate for missing modalities in a small portion of the population. Multimodal systems also have an advantage of lower error rates and higher accuracy when compared to unimodal systems . Consequently, multimodal systems have been deployed in many large scale biometric applications including the FBI’s Next Genration Identification (NGI), the Department of Homeland Security’s US-VISIT, and the Government of India’s UID. However, Multimodal systems have an increased demand for integrity and privacy because the system stores multiple biometric traits of each user. Hence, multimodal template protection is the main focus of this paper

The fundamental challenge in designing a biometric template protection scheme is to manage the intra-user variability that occurs due to signal variations in the multiple acquisitions of the same biometric trait. With respect to biometric template protection, four main architectures are widely used: fuzzy commitment, secure sketch, secure multiparty computation, and cancelable biometrics. Fuzzy commitment and secure sketch are biometric cryptosystem methods and are usually implemented with error correcting codes and provide information-theoretic guarantees of security and privacy. Secure multiparty computation architectures are distance based and use cryptographic tools. Cancelable biometrics use revocable and non-invertible user-specific transformations for distorting the enrollment biometric, with the matching typically performed in the transformed domain. For a template to be secure, it must satisfy the important.

For a template to be secure, it must satisfy the important properties of noninvertibility and revocability. Noninvertibility implies that given a template, it must be computationally difficult to recover the original biometric data from the template. Revocability implies that if a template gets compromised, it should be possible to revoke the compromised template and generate a new template using a different transformation Moreover, it should be difficult to identify that the new template and the old compromised template are generated from the same underlying biometric data.

One important issue for multimodal systems is that the multiple biometric traits generally do not have the same featurelevel representation. Furthermore, it is difficult to characterize multiple biometric traits using compatible feature-level representations, as required by a template protection scheme . To counter this issue there have been many fusion techniques for combining multiple biometrics. One possible approach is to apply a separate template protection scheme for each trait followed by decision-level fusion. However, such an approach may not be highly secure, since it is limited by the security of the individual traits. This issue motivated our proposed approach of using multimodal biometric security to perform a joint feature-level fusion and classification

Another important issue is that biometric cryptosystem schemes are usually implemented using error control codes. In order to apply error control codes, the biometric feature vectors must be quantized, for instance by binarizing. One method of binarizing the feature vectors is thresholding the feature vectors, for example, by thresholding against the population mean or thresholding against zero. However, thresholding causes a quantization loss and does not preserve the semantic properties of the data structure in Hamming space. In order to avoid thresholding and minimize the quantization loss, we have used the idea of hashing [15], [16], which is used in the image and data retrieval literature to achieve fast search by binarizing the real-valued image features. The basic idea of hashing is to map each visual object into a compact binary feature vector that approximately preserves the data structure in the original space. Owing to its storage and retrieval efficiency, hashing has been used for large scale visual search and image retrieval

Recent progress in image classification, object detection, face recognition, speech recognition and many other computer vision tasks demonstrates the impressive learning ability of convolutional neural networks (CNN). The robustness of features generated by the CNN has led to a surge in the application of deep learning for generating binary codes from raw image data. Deep hashing is the technique of integrating hashing and deep learning to generate compact binary vectors from raw image data. There is a rich literature related to the application of optimized deep learning for converting the raw image data to binary hash codes. Inspired by the recent success of deep hashing methods, the objective of this work is to examine the feasibility of integrating deep hashing with a secure architecture to generate a secure multimodal template for face and iris biometrics. Contributions include:

• We use deep hashing to generate a binary latent shared representation from a user’s face and iris biometrics.

• We combine cancelable biometrics and secure sketch schemes to create a hybrid secure architecture.

• We integrate the hybrid secure architecture with the deep hashing framework to generate a multimodal secure sketch, which is cryptographically hashed to generate the secure multimodal template.

• We analyze the trade-off between genuine accept rate (GAR) and security for the proposed secure multimodal scheme using an actual multimodal database.

Additionally, we also perform an information-theoretic privacy analysis, and unlinkability analysis for the proposed secure system.

The proposed approach represents a biometric security framework integrated with multimodal fusion and deep hashing, and is particularly well suited for structural data from modalities like face and iris. Our approach is not a general biometric security framework that can be applied to all biometric modalities, but rather a proposal that needs further study and validation.

**LITERATURE SURVEY:**

**Deep Learning**

Deep learning has emerged as a new area of machine learning and is being extensively applied to solve problems that have resisted the best attempts of the machine learning and artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business, and government

Deep learning has been extensively implemented and applied to image recognition tasks. Krizhevsky et al. provided a breakthrough in the field of object recognition and ImageNet classification by applying a CNN for object recognition. They were able to reduce the error rate by almost half. The neural network implemented in is currently known as AlexNet and triggered the rapid endorsement of deep learning by the computer vision community. Simonyan et al. increased the depth of the convolutional network but reduced the size of the filters being used for convolution. The main contribution in was a thorough evaluation of networks of increasing depth using an architecture with very small 3 × 3 convolution filters, which represented a compelling advancement over the prior-art configurations

Szegedy advanced the architecture of CNN by making it deeper, and wider by introducing a CNN termed inception. One particular incarnation of this architecture is known as GoogleNet which is 22 layers deep. He developed a very deep 152 layer convolutional neural network architecture named ResNet. The novelty of ResNet lies not only in creating a very deep network but also in the use of a residual architecture to reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions

In addition to improving performance in image and speech recognition, deep learning has produced extremely promising results for various tasks in natural language understanding, particularly topic classification, sentiment analysis, question answering, and language translation.

**Deep Hashing**

Many hashing methods have been proposed to enable efficient approximate nearest neighbor search due to low space and time complexity. These traditional hashing methods can be categorized into data-independent or datadependent methods. A comprehensive survey of hashing techniques is presented . Initial research on hashing was mainly focused on data-independent methods, such as locality sensitive hashing (LSH). LSH methods generate hashing bits by using random projections. However, LSH methods demand a significant amount of memory as they require long codes to achieve satisfactory performance.

To learn compact binary codes, data-dependent hashing methods have been proposed in the literature. Data-dependent methods learn similarity-preserving hashing functions from a training set. Data-dependent hashing methods can be categorized as unsupervised or supervised. These methods have achieved success to some extent by using handcrafted features for learning hash functions. However, the handcrafted features do not preserve the semantic data similarities of image pairs and non-linear variation in realworld data. This has led to a surge of deep hashing methods where deep neural networks encode non-linear hash functions. This leads to an effective end-to-end learning of feature representation and hash coding

Xia et al. [17] adopted a two-stage learning strategy wherein the first stage computes hash codes from the pairwise similarity matrix and the second stage trains a deep neural network to fit the hash codes generated in the first stage. The model proposed by Lai et al. [18] simultaneously captures the intermediate image features and trains the hashing function in a joint learning process. The hash function in [18] uses a divide-and-encode module, which splits the image features derived from the deep network into multiple blocks, each block encoded into one hash bit. Liu et al. [20] present a deep hashing model that learns the hash codes by simultaneously optimizing a contrastive loss function for input image pairs and imposing a regularization on the real-valued outputs to approximate the binary values. Zhu et al. [36] proposed a deep hashing method to learn hash codes by optimizing a pairwise cross-entropy quantization loss to preserve the pairwise similarity and minimize the quantization error simultaneously

Secure Biometrics

The leakage of biometric template information to an adversary constitutes a serious threat to security and privacy of the user because if an adversary gains access to the biometric database, he can potentially obtain the stored biometric information of a user. To alleviate the security and privacy concerns in biometric usage, secure biometric architectures have been developed to allow authentication without requiring that the reference biometric template be stored in its raw format at the access control device. Secure biometric architectures include biometric cryptosystems (e.g., fuzzy commitment and secure sketch) [4], [5], [7], [8] and transformation based methods (e.g., cancelable biometrics)

Fuzzy commitment, a classical method of biometric protection, was first proposed in 1999 [5]. Forward error correction (FEC) based fuzzy commitment can also be viewed as a method of extracting a secret code by means of polynomial interpolation [6]. An implementation example of such a fuzzy commitment scheme appears in [8], wherein a BCH code is employed for polynomial interpolation; experiments show that when the degree of the interpolated polynomial is increased, the matching becomes more stringent, reducing the false accept rate (FAR), but increasing the false reject rate (FRR). Cancelable biometrics was first proposed by Ratha et al. [9], after which, there have been various different methods of generating cancelable biometric templates. Some popular methods use non-invertible transforms [9], bio-hashing [10], salting [11] and random projections [12]. Literature surveys on cancelable biometrics can be found in [3], and [37].

Secure Multimodal Biometrics

The secure biometric frameworks have been extended to include multiple biometric traits of a user [1], [13], [14], [38]. In [13] face and fingerprint templates are concatenated to form a single binary string and this concatenated string is used as input to a secure sketch scheme. Kelkboom et al. [39] provided results for decision-level, feature-level, and score-level fusion of templates by using the number of errors corrected in a biometric cryptosystem as a measure of the matching score. Nagar et al. [1] developed a multimodal cryptosystem based on feature-level fusion using two different security architectures, fuzzy commitment and fuzzy vault. Fu et al. [40] theoretically analyzed four different versions of the multibiometric cryptosystem: no-split, MN-split, package, and biometric model, using template security and recognition accuracy as performance metrics. In the first three versions, the biometric templates are secured individually with a decision-level fusion, while the last version is a feature-level fusion.

Research has also been directed towards integrating cancelable biometric techniques into multimodal systems. Canuto et al. [38] combined voice and iris using cancelable transformations and decision level fusion. Paul and Gavrilova [41] used random projections and transformation-based feature extraction and selection to generate cancelable biometric templates for face and ear. There are some studies related to the use of multi-feature biometric fusion, which involves combining different features of the same biometric trait

However, none of the above papers present a secure architecture that combines multiple secure schemes to protect multiple biometrics of a user. In this paper, we have integrated a deep hashing framework with a hybrid secure architecture by combining cancelable biometric templates and secure sketch, which makes it computationally prohibitive to forge a combination of multiple biometrics that passes the authentication.

**3.3 SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

* System : i3 IV or Above.
* Hard Disk : 40 GB.
* Ram : 4 GB.

**SOFTWARE REQUIREMENTS:**

* **Operating System:** Windows 8 or Above
* **Coding Language**: Python 3.7

**3.4 SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

ECONOMICAL FEASIBILITY

TECHNICAL FEASIBILITY

SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### **TECHNICAL FEASIBILITY**

### This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**4.SYSTEM DESIGN**

**4.1 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:

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

**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.



**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

****

**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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

****

**COLLABORATION DIAGRAM:**



**4.3 IMPLEMENTATION:**

**MODULES:**

1. Deep Features Extraction:

using this module we will read finger and vein images and then extract pixels from both images and then convert those pixels into binary to generate a feature vector

1. CancelableModule:

using this module from each image we will select random bits and then this random selected bits will be input to VGG19 CNN model. CNN will use FUSION layer to combine both finger and vein features to generate raw features

1. Enrollment:

using this module VGG19 CNN model get trained on fusion features and then build a training model. This training model can be applied on test images to authenticate users

1. Authentication:

using this module application take two images as input (finger and vein) and then extract features from images and then apply VGG19 to predict and authenticate users

1. Multimodal CNN Graph:

To train VGG we took 20 iteration or epoch and using this module we will plot accuracy and loss graph of VGG at each epoch/iterations.

**5. SOFTWARE 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/" \t "_blank), 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**

**6.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**

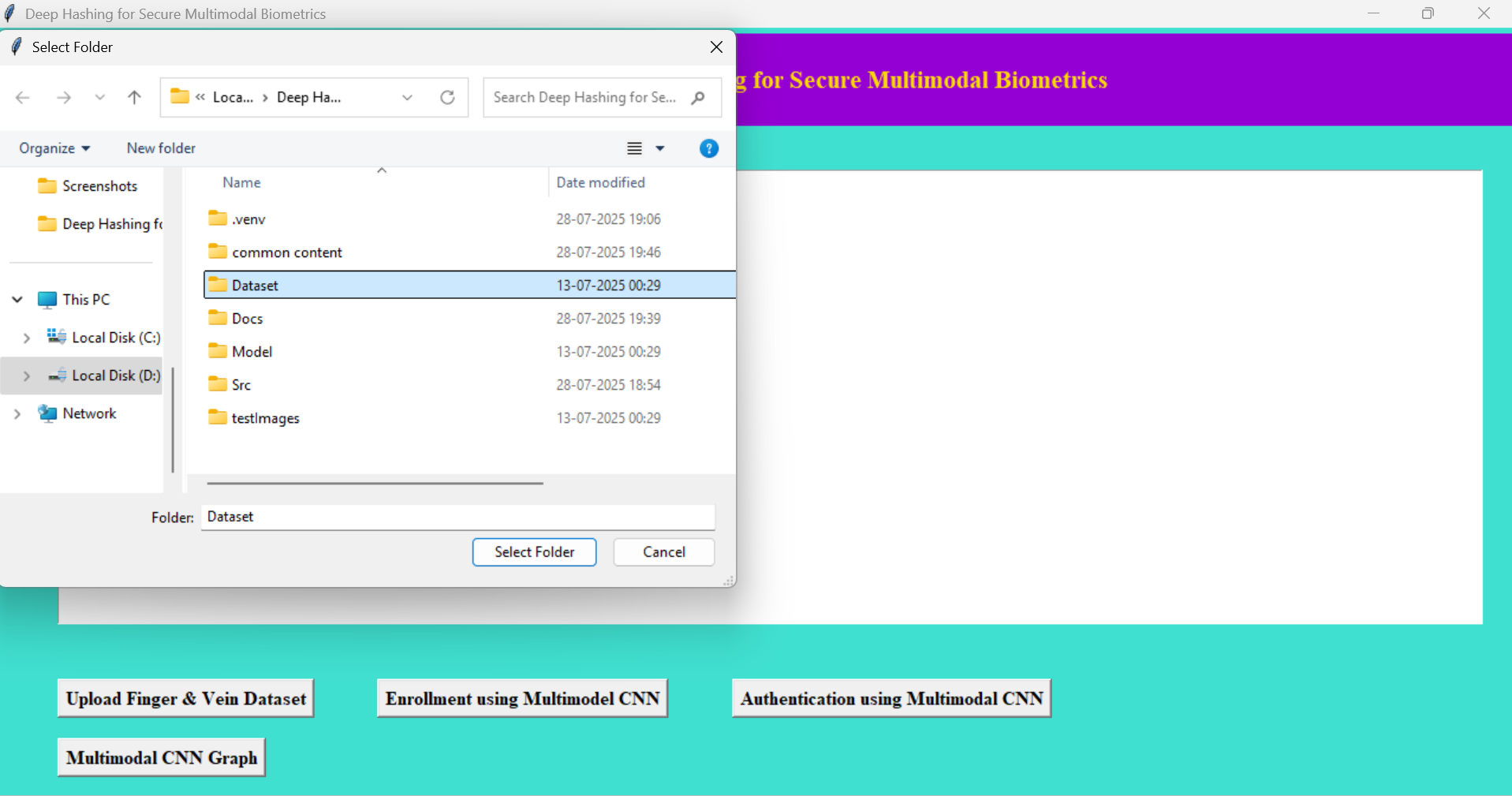
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

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

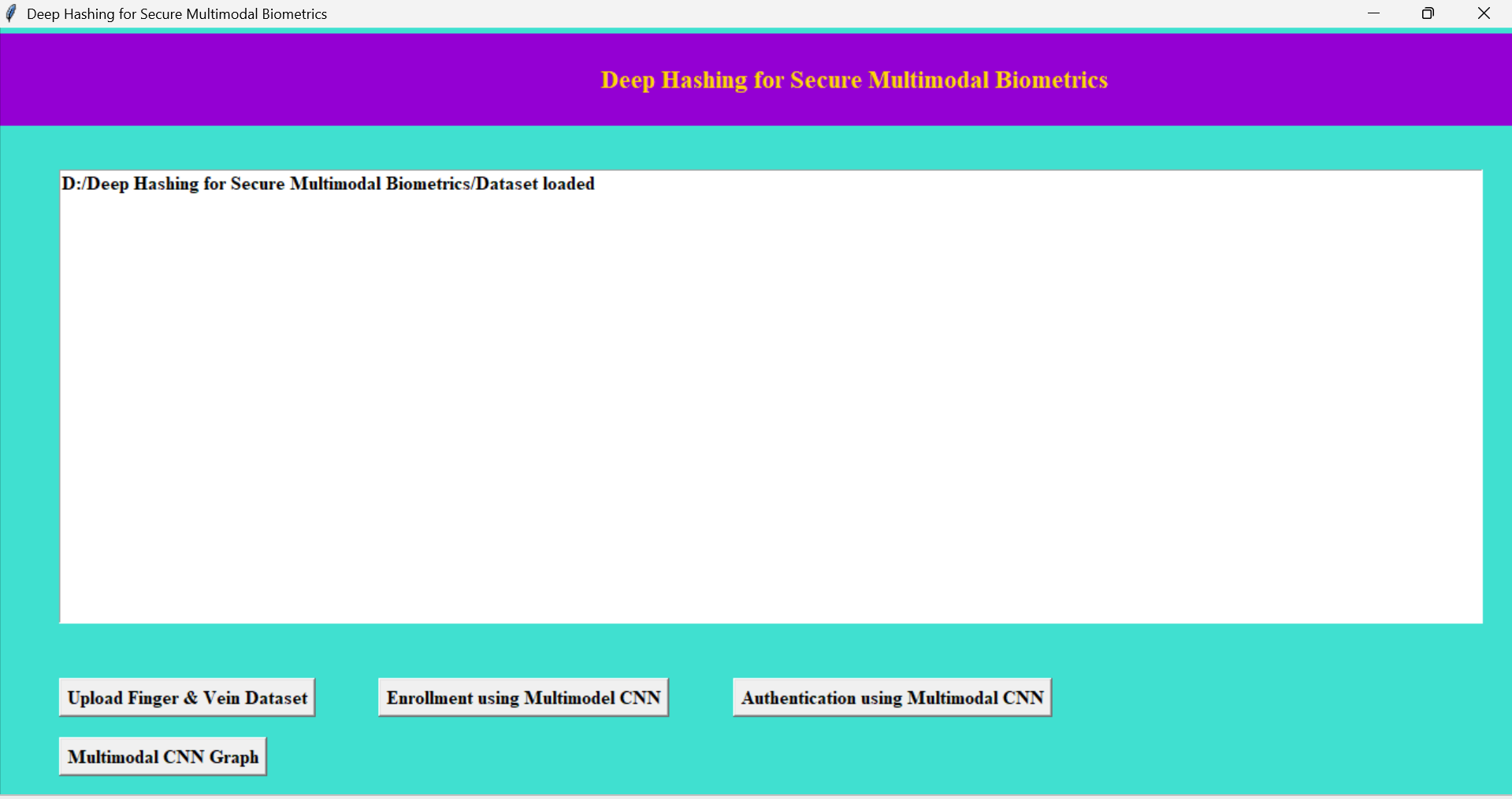
**SCREENSHOTS**



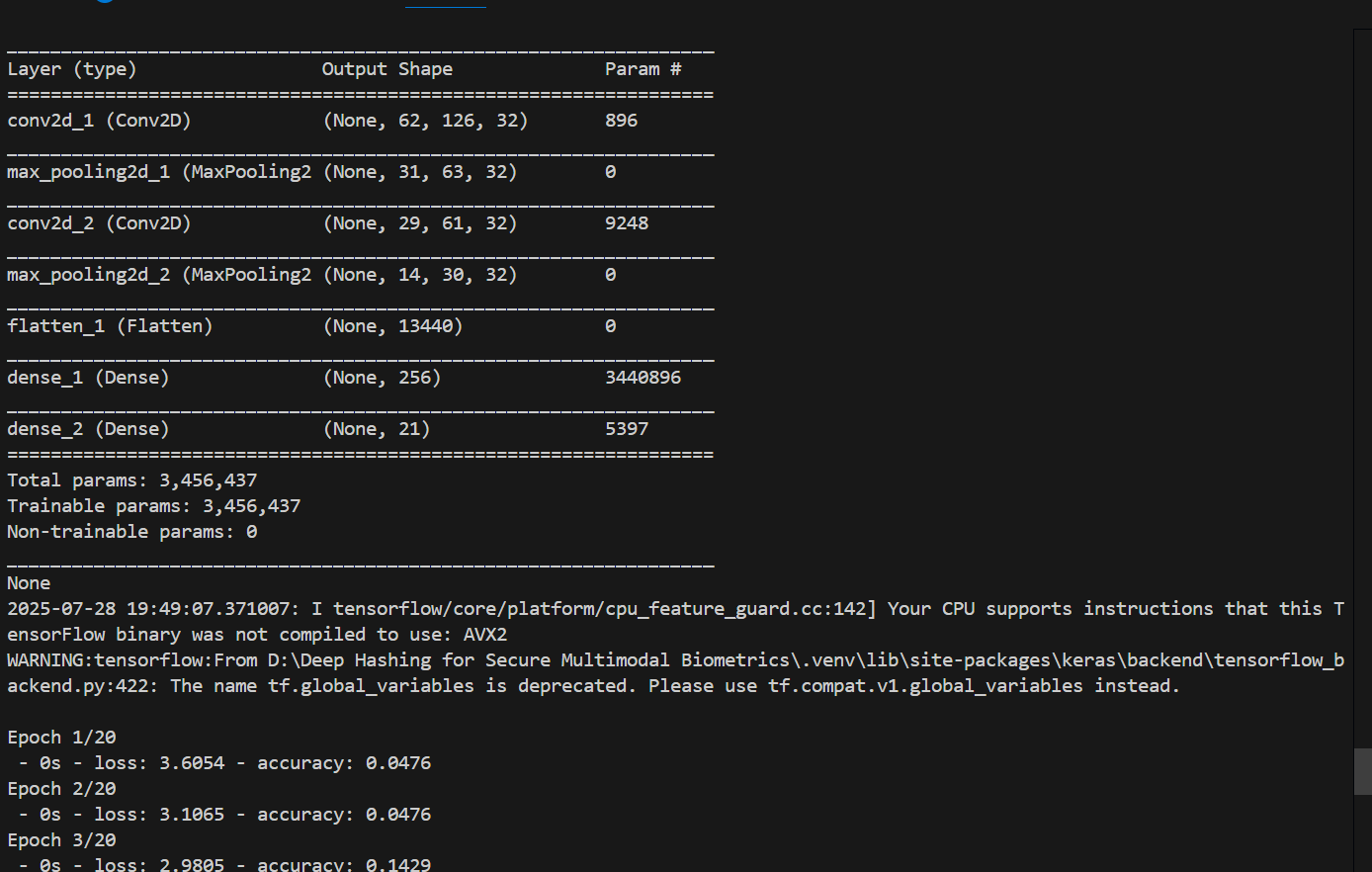
In above screen click on ‘Upload finger & Vein Dataset’ button to upload all 20 users dataset



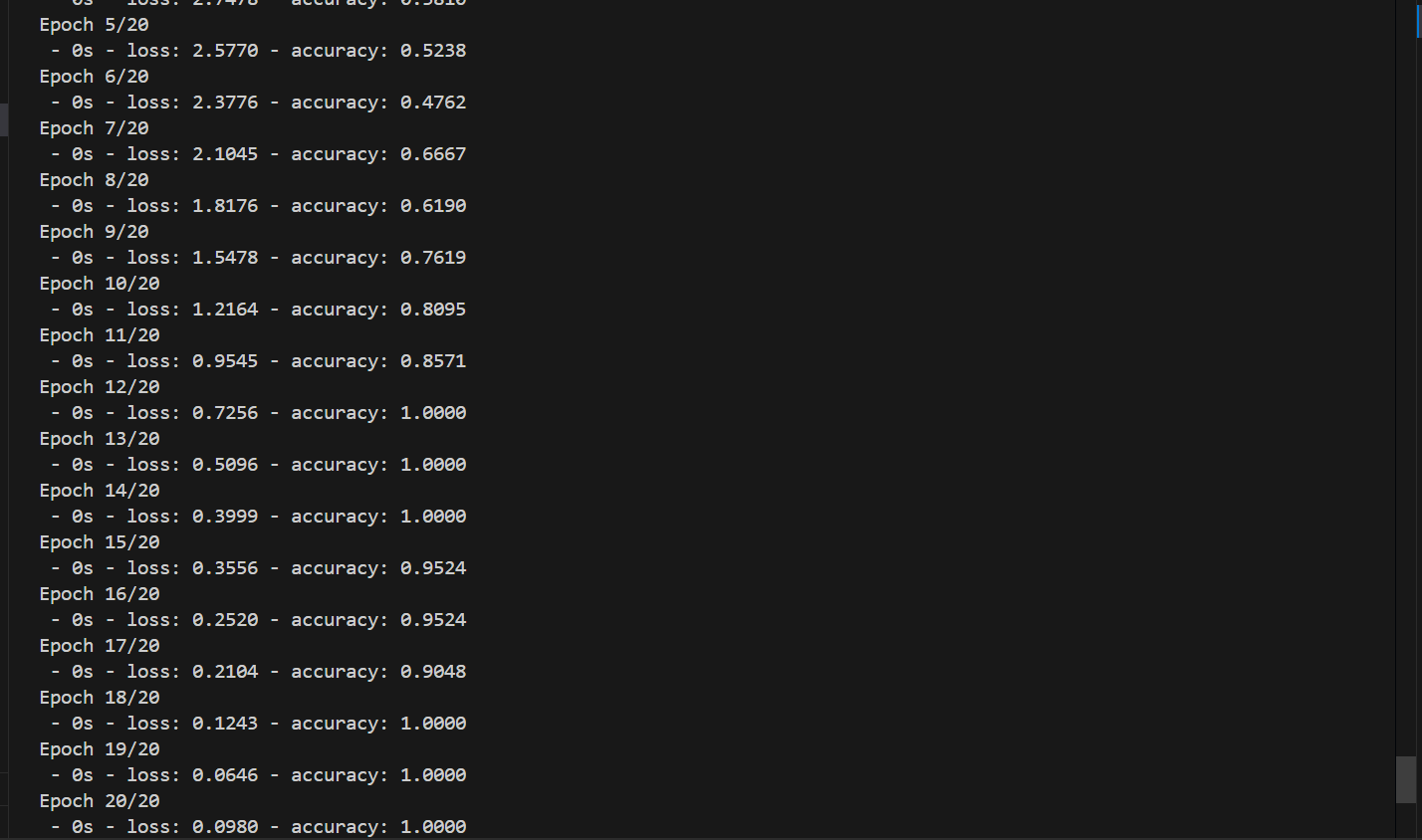
In above screen selecting and uploading entire ‘Dataset’ folder and then click on ‘Select Folder’ button to load dataset and to get below screen



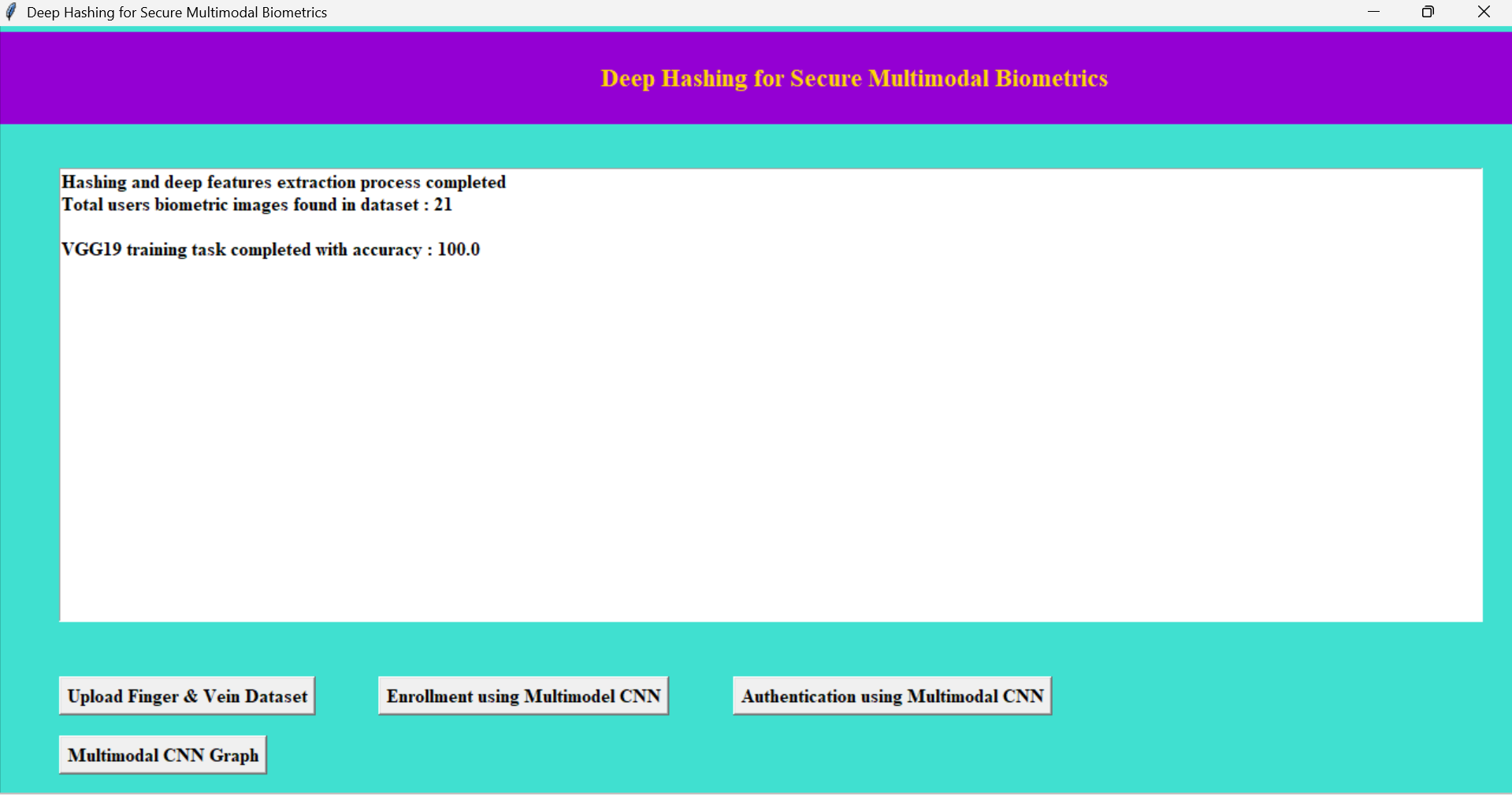
In above screen dataset loaded and now click on ‘Enrollment using Multimodal CNN’ button to extract features and then train CNN



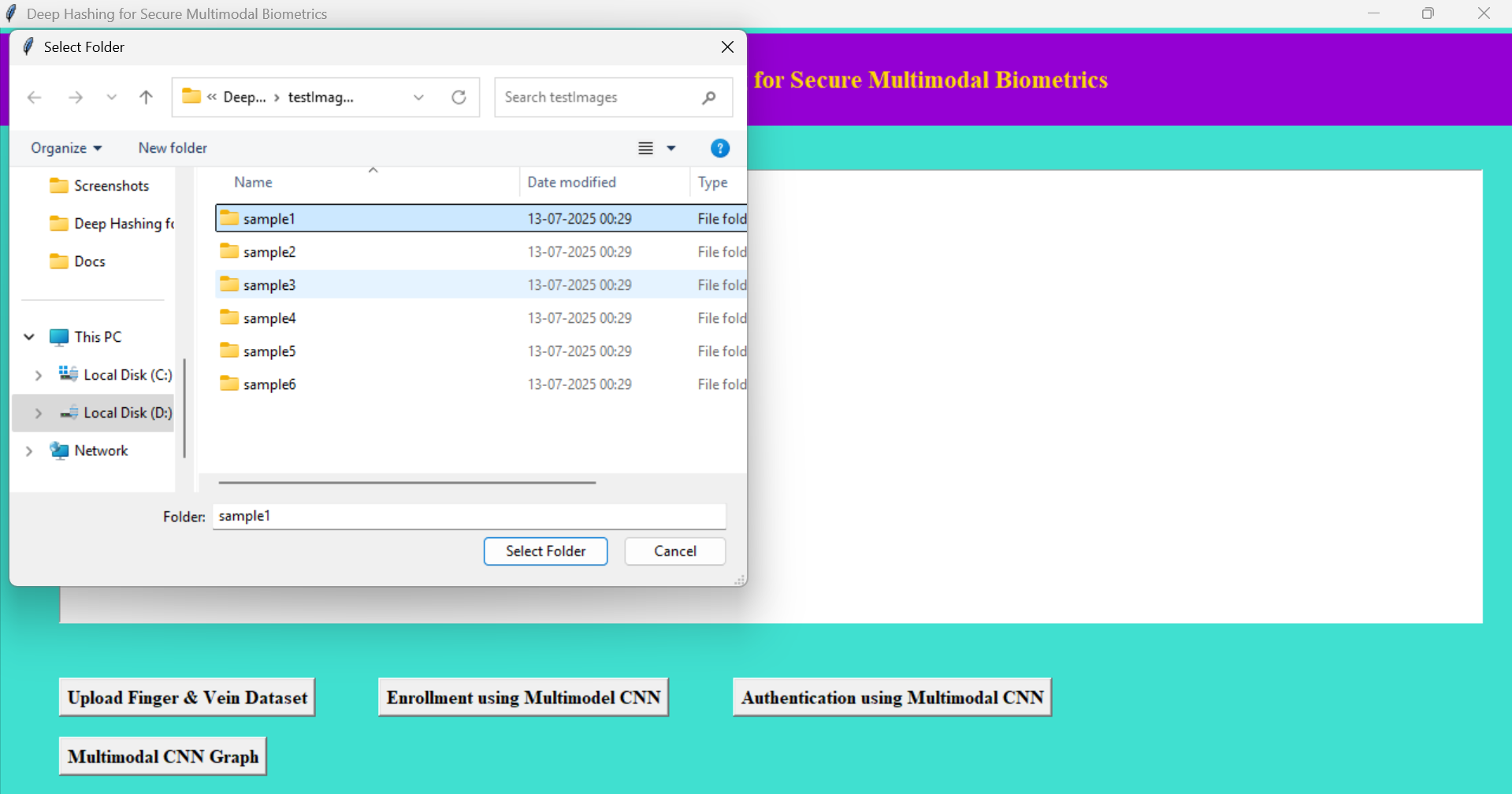
In above screen CNN model start generating and at each increasing epoch we can see VGG19 accuracy is getting increased and loss getting decreased



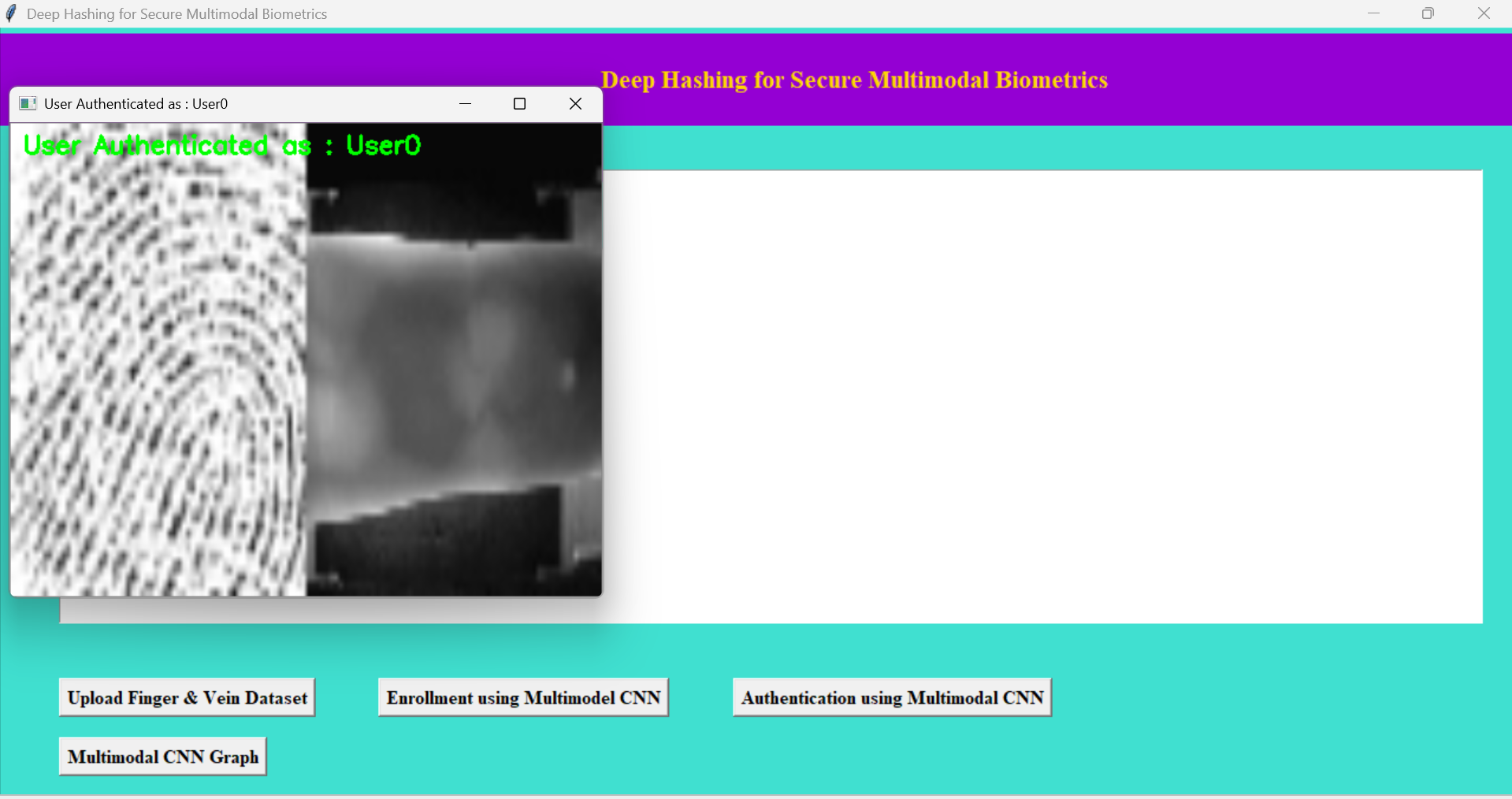
In above screen at final epoch we got VGG19 CNN accuracy as 100% and now see main application output in below screen



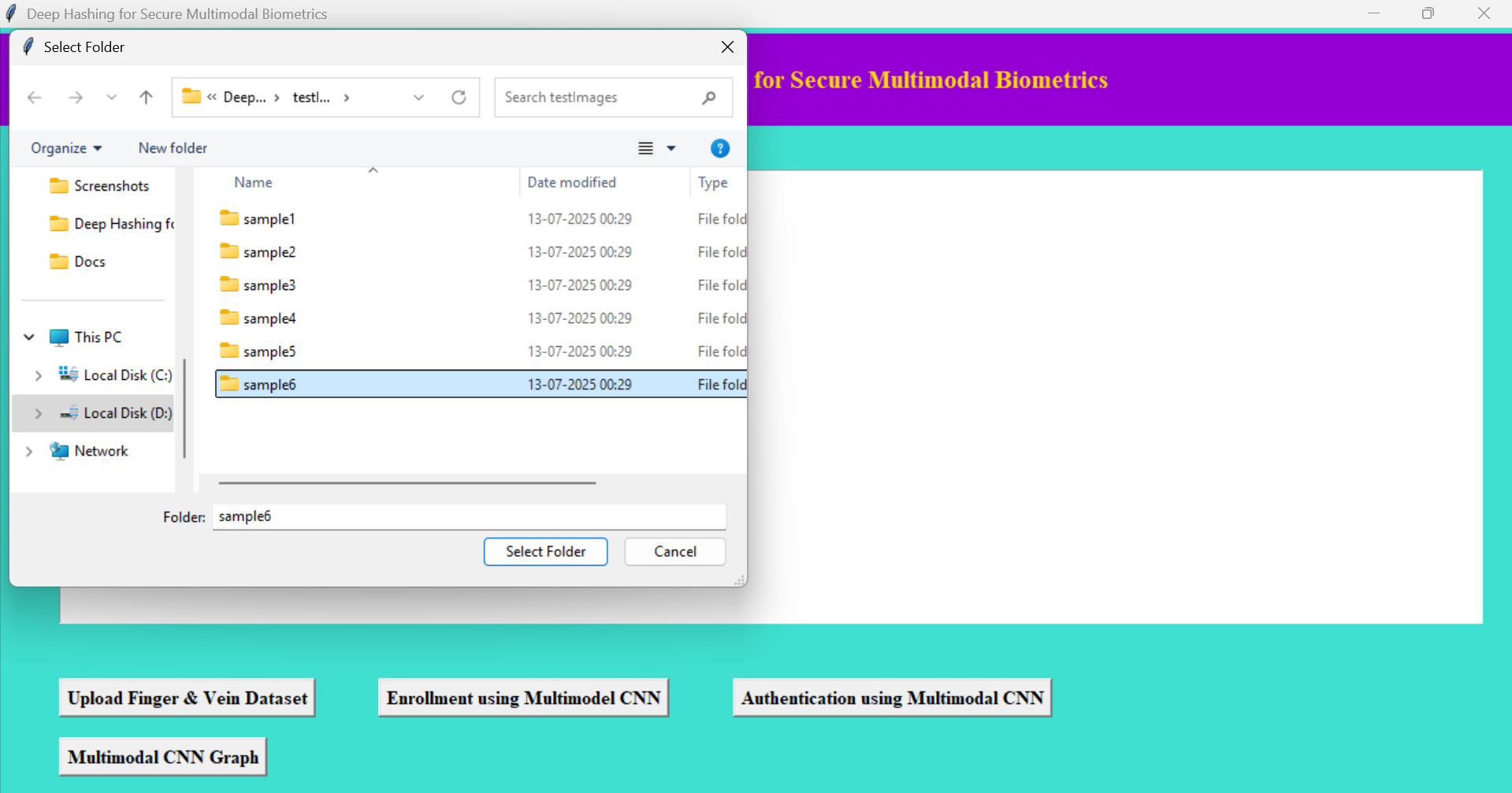
In above screen CNN got 21 users and after training all 21 users CNN got accuracy as 100%. Now click on ‘Authentication using Multimodal’ button to upload test images and then CNN will identify or authenticate users. Test images you can find inside ‘testImages’ folder

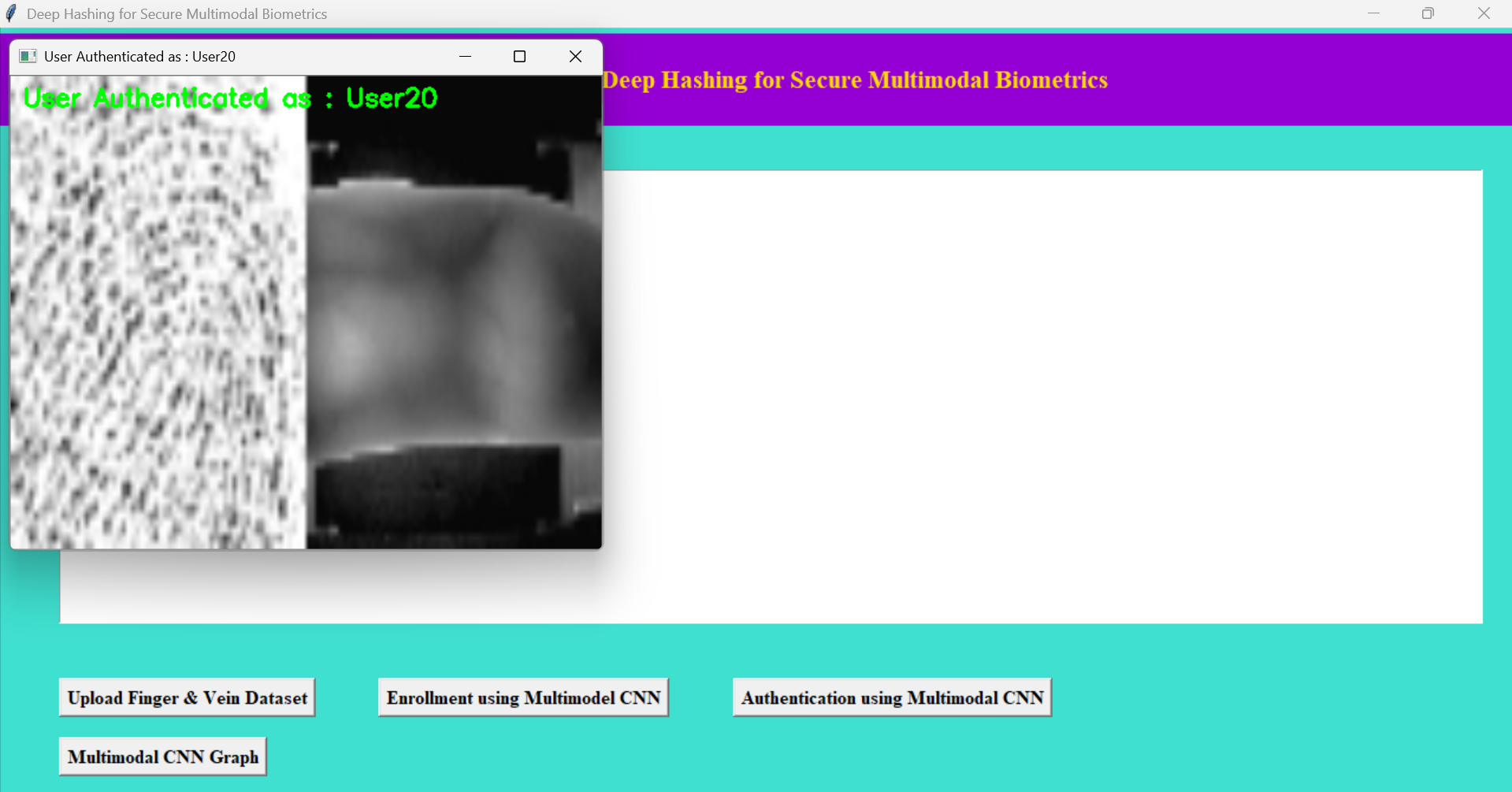


In above screen selecting and uploading ‘sample1’ folder and then click on ‘Select Folder’ button to get below result

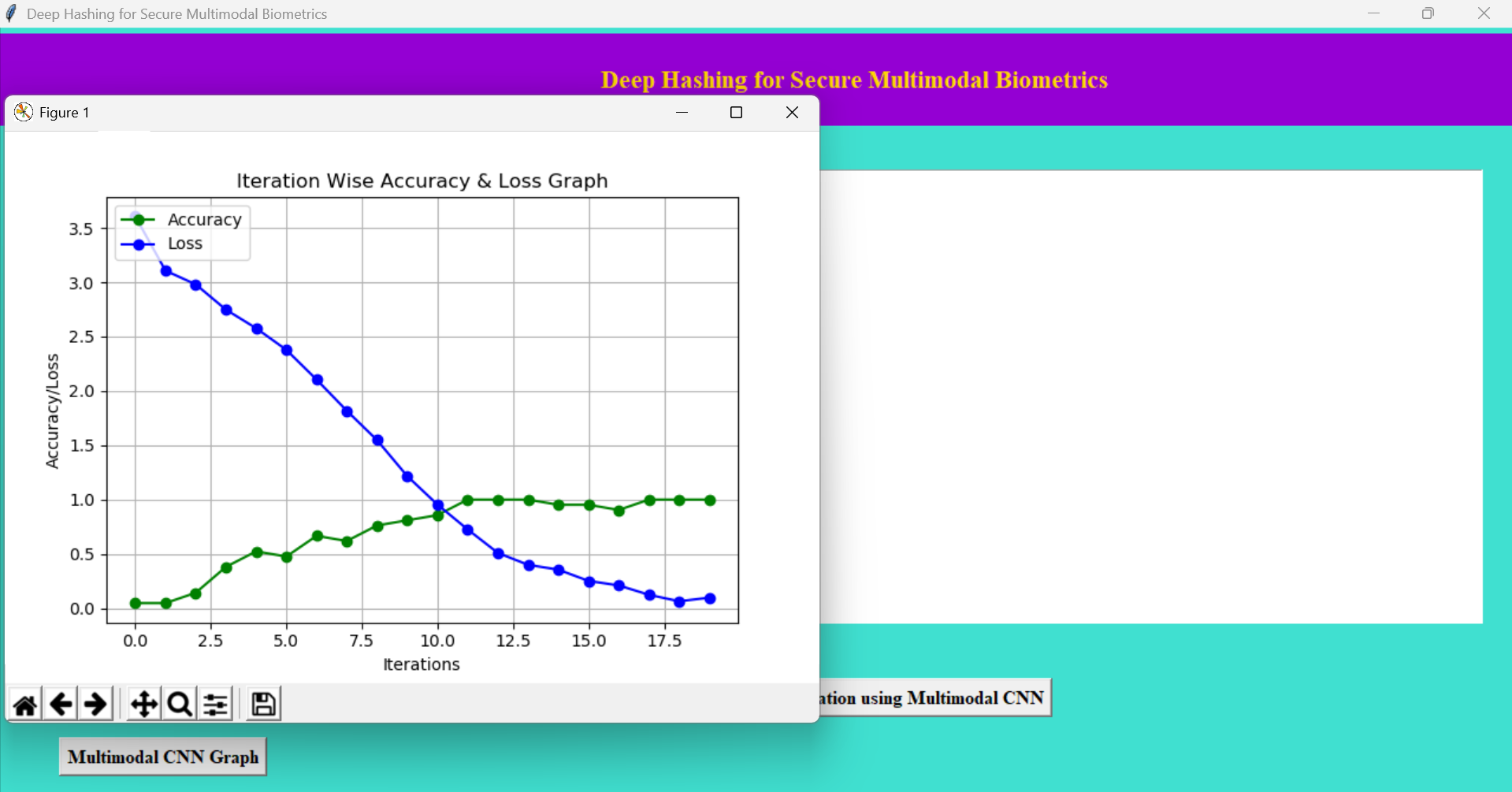


In above screen uploaded sample image identified as ‘User0’ and now try another image

in above screen uploading ‘sample6’ folder and below is the output



In above screen uploaded image authenticated as ‘User20’ and similarly you can upload and test other images. Now click on ‘Multimodal CNN Graph’ button to get below graph



In above graph x-axis represents EPOCH and y-axis represents accuracy and loss values and in above graph blue line represents loss and green line represents accuracy and in above graph we can see with each increasing epoch accuracy get increased and loss get decreased

**CONCLUSION**

We have presented a feature-level fusion and binarization framework using deep hashing to design a multimodal template protection scheme that generates a single secure template from each user’s multiple biometrics. We have employed a hybrid secure architecture combining the secure primitives of cancelable biometrics and secure-sketch and integrated it with a deep hashing framework, which makes it computationally prohibitive to forge a combination of multiple biometrics that passes the authentication. We have also proposed two deep learning based fusion architectures, fully connected architecture and bilinear architecture that could be used to combine more than two modalities. Moreover, we have analyzed the matching performance and the security, and also performed also unlinkability analysis of the proposed secure multibiometric system. Experiments using the WVU multimodal dataset, which contain face and iris modalities, demonstrate that the matching performance does not deteriorate with the proposed protection scheme. In fact, both the matching performance and the template security are improved when using the proposed secure multimodal system. However, we want to clarify that while the proposed solution is an interesting biometric security framework, in particular for structured data from modalities like face and iris, further validation is required to show how much it can work with other biometric modalities. Finally, the goal of this paper is to motivate researchers to investigate how to generate secure compact multimodal templates.

**REFERENCES**

[1] A. Nagar, K. Nandakumar, and A. K. Jain, “Multibiometric cryptosystems based on feature-level fusion,” IEEE Trans. Inf. Forensics Security, vol. 7, no. 1, pp. 255–268, Feb. 2012.

[2] A. Ross and A. K. Jain, “Multimodal biometrics: An overview,” in Proc. 12th Eur. Signal Process. Conf. (EUSIPCO), Sep. 2004, pp. 1221–1224.

[3] S. Rane, Y. Wang, S. C. Draper, and P. Ishwar, “Secure biometrics: Concepts, authentication architectures, and challenges,” IEEE Signal Process. Mag., vol. 30, no. 5, pp. 51–64, Sep. 2013.

[4] Y. Sutcu, Q. Li, and N. Memon, “Protecting biometric templates with sketch: Theory and practice,” IEEE Trans. Inf. Forensics Security, vol. 2, no. 3, pp. 503–512, Sep. 2007.

[5] A. Juels and M. Wattenberg, “A fuzzy commitment scheme,” in Proc. 6th ACM Conf. Comput. Commun. Secur. (CCS), Nov. 1999, pp. 28–36.

[6] A. Juels and M. Sudan, “A fuzzy vault scheme,” in Proc. IEEE Int. Symp. Inf. Theory, Jul. 2002, p. 408.

[7] K. Nandakumar, A. K. Jain, and S. Pankanti, “Fingerprint-based fuzzy vault: Implementation and performance,” IEEE Trans. Inf. Forensics Security, vol. 2, no. 4, pp. 744–757, Dec. 2007.

[8] A. Nagar, K. Nandakumar, and A. K. Jain, “Securing fingerprint template: Fuzzy vault with minutiae descriptors,” in Proc. 19th Int. Conf. Pattern Recognit., Dec. 2008, pp. 1–4.

[9] N. K. Ratha, S. Chikkerur, J. H. Connell, and R. M. Bolle, “Generating cancelable fingerprint templates,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 4, pp. 561–572, Apr. 2007.

[10] A. Kong, K.-H. Cheung, D. Zhang, M. Kamel, and J. You, “An analysis of BioHashing and its variants,” Pattern Recognit., vol. 39, no. 7, pp. 1359–1368, Jul. 2006.

[11] J. Zuo, N. K. Ratha, and J. H. Connell, “Cancelable iris biometric,” in Proc. 19th Int. Conf. Pattern Recognit., Dec. 2008, pp. 1–4.

[12] A. B. J. Teoh, Y. W. Kuan, and S. Lee, “Cancellable biometrics and annotations on BioHash,” Pattern Recognit., vol. 41, no. 6, pp. 2034–2044, Jun. 2008.

[13] Y. Sutcu, Q. Li, and N. Memon, “Secure biometric templates from fingerprint-face features,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2007, pp. 1–6.

[14] K. Nandakumar and A. K. Jain, “Multibiometric template security using fuzzy vault,” in Proc. IEEE 2nd Int. Conf. Biometrics, Theory, Appl. Syst., Sep. 2008, pp. 1–6.

[15] A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing,” in Proc. Int. Conf. Very Large Data Bases, vol. 99, no. 6, 1999, pp. 518–529.

[16] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, “Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[17] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, “Supervised hashing for image retrieval via image representation learning,” in Proc. AAAI Conf. Artif. Intell., Jul. 2014, p. 2.

[18] H. Lai, Y. Pan, Y. Liu, and S. Yan, “Simultaneous feature learning and hash coding with deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 3270–3278.

[19] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, “Learning compact binary descriptors with unsupervised deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 1183–1192.

[20] H. Liu, R. Wang, S. Shan, and X. Chen, “Deep supervised hashing for fast image retrieval,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 2064–2072.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., Dec. 2012, pp. 1097–1105.

[22] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” Sep. 2014, arXiv:1409.1556. [Online]. Available: <https://arxiv.org/abs/1409.1556>

[23] C. Szegedy et al., “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1–9.

[24] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[25] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Learning hierarchical features for scene labeling,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[26] K. He, F. Wen, and J. Sun, “K-means hashing: An affinity-preserving quantization method for learning binary compact codes,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2013, pp. 2938–2945.

[27] P. Jain, B. Kulis, and K. Grauman, “Fast image search for learned metrics,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2008, pp. 1–8.

[28] B. Kulis and T. Darrell, “Learning to hash with binary reconstructive embeddings,” in Proc. Neural Inf. Process. Syst. (NIPS), Dec. 2009, pp. 1042–1050.

[29] M. Norouzi and D. M. Blei, “Minimal loss hashing for compact binary codes,” in Proc. 28th Int. Conf. Mach. Learn., Jul. 2011, pp. 353–360.

[30] M. Raginsky and S. Lazebnik, “Locality-sensitive binary codes from shift-invariant kernels,” in Proc. Adv. Neural Inf. Process. Syst., Dec. 2009, pp. 1509–1517.

[31] J. Wang, H. T. Shen, J. Song, and J. Ji, “Hashing for similarity search: A survey,” Aug. 2014, arXiv:1408.2927. [Online]. Available: <https://arxiv.org/abs/1408.2927>

[32] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), Dec. 2009, pp. 1753–1760.

[33] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, “Content-based image retrieval at the end of the early years,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 12, pp. 1349–1380, Dec. 2000.

[34] X. Yuan, L. Ren, J. Lu, and J. Zhou, “Relaxation-free deep hashing via policy gradient,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Sep. 2018, pp. 134–150.

[35] Z. Chen, X. Yuan, J. Lu, Q. Tian, and J. Zhou, “Deep hashing via discrepancy minimization,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 6838–6847.

[36] H. Zhu, M. S. Long, J. M. Wang, and Y. Cao, “Deep hashing network for efficient similarity retrieval,” in Proc. 13th Conf. Amer. Assoc. Artif. Intell., Feb. 2016, pp. 2415–2421.

[37] V. M. Patel, N. K. Ratha, and R. Chellappa, “Cancelable biometrics: A review,” IEEE Signal Process. Mag., vol. 32, no. 5, pp. 54–65, Sep. 2015.

[38] A. M. P. Canuto, F. Pintro, and J. C. Xavier-Junior, “Investigating fusion approaches in multi-biometric cancellable recognition,” Expert Syst. Appl., vol. 40, no. 6, pp. 1971–1980, May 2013.

[39] E. J. C. Kelkboom, X. Zhou, J. Breebaart, R. N. J. Veldhuis, and C. Busch, “Multi-algorithm fusion with template protection,” in Proc. IEEE 3rd Int. Conf. Biometrics, Theory, Appl., Syst., Sep. 2009.

[40] B. Fu, S. X. Yang, J. Li, and D. Hu, “Multibiometric cryptosystem: Model structure and performance analysis,” IEEE Trans. Inf. Forensics Security, vol. 4, no. 4, pp. 867–882, Dec. 2009.

[41] P. P. Paul and M. Gavrilova, “Multimodal cancelable biometrics,” in Proc. IEEE 11th Int. Conf. Cognit. Informat. Cognit. Comput., Aug. 2012, pp. 43–49.

[42] C. Rathgeb and C. Busch, “Cancelable multi-biometrics: Mixing iriscodes based on adaptive Bloom filters,” Comput. Secur., vol. 42, pp. 1–12, May 2014.

[43] O. Russakovsky et al., “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, Dec. 2015.

[44] K. D. Nguyen, C. Fookes, A. Ross, and S. Sridharan, “Iris recognition with off-the-shelf CNN features: A deep learning perspective,” IEEE Access, vol. 6, pp. 18848–18855, 2017.

[45] Z. Zhao and A. Kumar, “Towards more accurate iris recognition using deeply learned spatially corresponding features,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 3809–3818.

[46] S. Minaee, A. Abdolrashidiy, and Y. Wang, “An experimental study of deep convolutional features for iris recognition,” in Proc. IEEE Signal Process. Med. Biol. Symp. (SPMB), Dec. 2016, pp. 1–6.

[47] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 815–823.

[48] Y. Sun, D. Liang, X. Wang, and X. Tang, “DeepID3: Face recognition with very deep neural networks,” Feb. 2015, arXiv:1502.00873. [Online]. Available: <https://arxiv.org/abs/1502.00873>

[49] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” in Proc. Brit. Mach. Vis. Conf. (BMVC), Sep. 2015, pp. 41.1–41.12.

[50] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Convolutional two-stream network fusion for video action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 1933–1941.

[51] T.-Y. Lin, A. RoyChowdhury, and S. Maji, “Bilinear CNN models for fine-grained visual recognition,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1449–1457.

[52] Z. Cao, M. Long, J. Wang, and P. S. Yu, “HashNet: Deep learning to hash by continuation,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 5609–5618.

[53] Y. Sutcu, S. Rane, J. S. Yedidia, S. C. Draper, and A. Vetro, “Feature extraction for a slepian-wolf biometric system using LDPC codes,” in Proc. IEEE Int. Symp. Inf. Theory, Jul. 2008, pp. 2297–2301.

[54] D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Learning face representation from scratch,” Nov. 2014, arXiv:1411.7923. [Online]. Available: <https://arxiv.org/abs/1411.7923>

[55] WVU Multimodal Dataset. Accessed: Jan. 28, 2018. [Online]. Available: <http://biic.wvu.edu/>

[56] D. E. King, “Dlib-ML: A machine learning toolkit,” J. Mach. Learn. Res., vol. 10, pp. 1755–1758, Jan. 2009.

[57] CASIA-Iris-Thousand. Accessed: Dec. 2, 2017. [Online]. Available: <http://biometrics.idealtest.org/>

[58] K. W. Bowyer and P. J. Flynn, “The ND-IRIS-0405 iris image dataset,” CVRL, Dept. Comp. Sci. Eng., Univ Notre Dame, Notre Dame, IN, USA, Tech. Rep., 2010.

[59] G. Sutra, B. Dorizzi, S. Garcia-Salitcetti, and N. Othman, “A biometric reference system for iris (OSIRIS),” Telecom SudParis, Paris, France, Tech. Rep., 2013.

[60] V. Talreja, M. C. Valenti, and N. M. Nasrabadi, “Multibiometric secure system based on deep learning,” in Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP), Nov. 2017, pp. 298–302.

[61] H.-F. Yang, K. Lin, and C.-S. Chen, “Supervised learning of semanticspreserving hash via deep convolutional neural networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 2, pp. 437–451, Feb. 2018.

[62] Information Technology—Security Techniques—Biometric Information Protection, Standard ISO/IEC 24745:2011, ISO/IEC JTC1 SC27 Security Techniques, 2011.

[63] M. Gomez-Barrero, J. Galbally, C. Rathgeb, and C. Busch, “General framework to evaluate unlinkability in biometric template protection systems,” IEEE Trans. Inf. Forensics Security, vol. 13, no. 6, pp. 1406–1420, Jun. 2018.