A PROJECT REPORT ON

FABRIC AUTHENTICATION SYSTEM USING DEEP LEARNING

Submitted in partial fulfilment of the award of
Post Graduate Diploma in Big Data Analytics
(PG-DBDA) from
Authorized Training Centre

Under the Guidance of
Mr Sadhu Sreenivas Sir

Submitted by:

Aniket Udhav Mungare	**PRN No: 230950325003**
Ajay Ramesh Bet	**PRN No: 230950325009**
Chandu Satya Priyanka	**PRN No: 230950325010**
Hrishikesh S Jamkhande	**PRN No: 230950325016**
Omkar Sunil Khurd	**PRN No: 230950325020**
Utkarsha Mushre	**PRN No: 230950325029**

CERTIFICATE

This is to certify that the project work under the title **"FABRIC AUTHENTICATION SYSTEM USING DEEP LEARNING"** is done by

Mr Aniket Udhav Mungare	**Mr Hrishikesh S Jamkhande**
Mr Ajay Ramesh Bet	**Mr Omkar Sunil Khurd**
Ms Chandu Satya Priyanka	**Ms Utkarsha Mushre**

in partial fulfilment of the requirement for award of **Diploma in Big Data Analytics** Course.

Project Guide	**Course Coordinator**
Mr Sadhu Srinivas sir	Mr Sadhu Srinivas sir

Date:

Location: CDAC ACTS Hyderabad.

ACKNOWLEDGEMENT

This project "**FABRIC AUTHENTICATION BY USING DEEP LEARNING** " was a great learning experience for us and we are submitting this work to Centre for Development of Advance Computing (CDAC).

We are very glad to mention ***Mr Sadhu Srinivas Sir*** for valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

Sr.No	Name	PRN No	Sign
1	Aniket Udhav Mungare	230950325003	
2	Ajay Ramesh Bet	230950325009	
3	Chandu Satya Priyanka	230950325010	
4	Hrishikesh S Jamkhande	230950325016	
5	Omkar Sunil Khurd	230950325020	
6	Utkarsha Mushre	230950325029	

INDEX

ABSTRACT

The texture of fabric is considered to be an important factor of the design and prediction of Machine-made fabric and handmade fabric. Traditionally, the recognition of fabric has a lot of challenges due to its manual visual inspection. Moreover, the approaches based on early machine learning algorithms directly depend on handcrafted features, which are time-consuming and error-prone processes. Hence, an AI system is needed for classification of fabric in Machine-made and Handmade.

In this Project Report, we propose a deep learning model based on data augmentation and learning approach for the classification and recognition of Difference between Machine-made and Handmade Fabric. The fabric images are enhanced by pre-processing at various levels using conventional image processing techniques and they are used to train the networks. The model uses the residual network **(VGG16).**We evaluated the results of our model using evaluation metrics such as accuracy, balanced accuracy, and F1-score. With the Fabric dataset, a maximum classification accuracy of 80**%** is achieved in the conducted simulations. The experimental results show that the proposed model is robust and achieves state-of-the-art accuracy even when the physical properties of the fabric are changed. We compared our results with other baseline approaches and a pertained **Resnet50** deep learning model which showed that the proposed method achieved higher accuracy.

Also, we create a Web Application of this model on Streamlit which is open-source framework to rapidly build web apps. It is a Python-based library specifically designed for machine learning engineers.

PROJECT OVERVIEW

1. Purpose:

The purpose of this project extends beyond theoretical exploration it addresses real-world applications with practical implications across various industries. By employing a deep learning model like VGG16 to distinguish between machine-made and handmade fabrics with high accuracy, coupled with the development of a user-friendly web application using Streamlit.

The ability to quickly and accurately identify the type of fabric, whether machine-made or handmade, is crucial for quality control in textile manufacturing and User. This project offers a reliable automated solution to assess fabric type, ensuring consistency and meeting industry standards.

Handmade fabrics often traditional craftsmanship and cultural heritage. By showcasing the unique characteristics of handmade fabrics through this project, artisans and craftsmen can gain recognition and appreciation, potentially leading to increased demand and support for their work.

This project can serve as an educational tool for textile students, researchers, and professionals to understand the between different types of fabrics. It can also be integrated into training programs for textile inspectors and quality control personnel to enhance their expertise in fabric assessment.

By accurately identifying handmade fabrics, businesses can cater to this growing market segment and promote sustainability initiatives within the textile industry.

2. Scope:

The scope of this project encompasses several key components, each contributing to the overreaching goal of distinguishing between machine-made and handmade fabrics using a deep learning model (VGG16) and a web application (Streamlit).

- **Data Collection and Pre-processing**: Gathering a diverse dataset comprising images of machine-made and handmade fabrics is essential for model training. This involves sourcing images from various sources, ensuring sufficient diversity in fabric types, textures, and patterns. Pre-processing tasks such as resizing, normalization, and augmentation may be required to enhance dataset quality and balance.

- **Model Development and Training**: Implementing the VGG16 deep learning architecture involves developing the model architecture, configuring parameters, and compiling the model. Training the model on the prepared dataset involves feeding the images.
- **Validation and Evaluation**: Assessing the model's performance involves validation and evaluation steps. Validation techniques such as cross-validation may be employed to ensure model generalization. Evaluation metrics such as accuracy, precision, recall, and F1-score are used to quantify the model's effectiveness in distinguishing between machine-made and handmade fabrics.
- **Web Application Development**: Creating a user-friendly web application using Streamlit involves designing the interface, integrating the trained model, and implementing interactive features for users to upload images and receive real-time predictions. The application should provide visual feedback and explanations to enhance user understanding of the model's predictions.
- **Testing and Deployment**: Testing the integrated system involves thorough validation of the web application's functionality, including model inference accuracy, responsiveness, and user experience. Deployment encompasses hosting the web application on a server, ensuring scalability, security, and accessibility for users to access the application remotely.

The scope of this project is focused on leveraging deep learning and web technologies to build a robust system for differentiating between machine-made and handmade fabrics, with potential applications.

3. Technologies used

- Python Framework
- Tensor flow, Keras libraries
- Streamlit (Python based Library)
- Github
- AWS Cloud Service

INTRODUCTION

The textile industry has witnessed significant advancements in recent years, with the introduction of machine-made fabrics revolutionizing the production process. However, amidst this technological progress, the timeless art of handmade fabric creation continues to hold a distinct attraction. This project report explores the Entire differences between machine-made and handmade fabrics, utilizing a deep learning model, specifically VGG16, to discern and classify these variations with an impressive accuracy of **97%**.

In this report, we delve into the methodologies employed to develop and train the **VGG16** model, leveraging its deep convolutional neural network architecture to analyse complex patterns, textures, and structural distinction inherent in both machine-made and handmade fabrics. Furthermore, we discuss the utilization of a web application built using Streamlit, enabling seamless interaction with the model for visualizing and comprehending the inequality between these fabric types.

Through diligent data collection, pre-processing, and model training, we demonstrate the strength of our approach in accurately discriminating between machine-made and handmade fabrics, shedding light on the distinctive attributes that distinguish one from the other.

This project combines cutting-edge deep learning techniques with user-friendly web applications, fostering a comprehensive understanding of the intricate interplay between tradition and technology in the sector of fabric production.

As we navigate through the intricacies of machine-made and handmade fabrics, this report serves as a testament to the power of artificial intelligence in solving the delicacy that defines our material world.

Formation of Fabric

1. Machine-made Fabric/Power loom Fabric:

First, the fibres (like cotton or polyester) are cleaned and stretched into thin strands called threads. These threads are spun together using machines to make them stronger and longer. The threads are then woven together on large machines to make a big piece of fabric. This can also be done by knitting the threads together, like how you might knit a scarf. Finally, the fabric goes through processes like washing, dyeing, and sometimes adding special treatments to make it look and feel just right.

2. Hand-made fabric:

Similar to machine-made fabric, fibres are cleaned and stretched into threads, but this is done by hand using tools like spinning wheels. Instead of using big machines, people weave the threads together on smaller looms or knit them with needles, using their hands to create the fabric. Because it's done by hand, each piece of handmade fabric can be unique, with its own little imperfections and special characteristics. Handmade fabric may still go through washing, dyeing, and other finishing touches, but these are often done with more care and attention to detail.

Difference between Machine-made Fabric and Handmade Fabric

1. **Production Process**:
   - **Machine-Made Fabric**: Produced using automated machines in a factory setting, involving standardized processes for spinning, weaving, or knitting.
   - **Handmade Fabric:** Created manually by artisans using traditional methods, often involving hand spinning, weaving, or knitting.

2. **Consistency and Uniformity**:
   - **Machine-Made Fabric**: Exhibits a high level of consistency and uniformity in texture, thickness, and appearance due to automated production processes.
   - **Handmade Fabric:** Tends to have more variability in texture, thickness, and appearance due to the manual nature of production, resulting in a unique and often irregular aesthetic.

3. **Time and Labor**:
   - **Machine-Made Fabric**: Requires less time and labour to produce large quantities of fabric due to automation, making it more cost-effective for mass production.
   - **Handmade Fabric:** Involves a significant amount of time and labour as each piece is crafted individually, often making it more expensive and exclusive.

4. **Customization and Artistry**:
   - **Machine-Made Fabric**: Offers limited scope for customization and individual artistry as production is standardized and mass-produced.
   - **Handmade Fabric:** Allows for greater customization and artistic expression as artisans can manipulate the materials and techniques to create unique designs and textures.

5. **Quality and Durability**:
    - **Machine-Made Fabric:** Generally exhibits consistent quality and durability suitable for a wide range of applications, including mass-produced clothing and furnishings.
    - **Handmade Fabric:** Quality and durability may vary depending on the skill of the artisan and the materials used, often prized for its craftsmanship and unique characteristics.

6. **Environmental Impact**:
    - **Machine-Made Fabric:** May have a larger environmental footprint due to the energy and resources required for industrial production, as well as potential waste from mass manufacturing.
    - **Handmade Fabric:** Often considered more environmentally friendly as it may use sustainable materials and techniques, and production processes are typically less energy-intensive.

While machine-made fabric offers efficiency and consistency suitable for large-scale production, handmade fabric is valued for its craftsmanship, uniqueness, and personalized touch, often appealing to individuals seeking distinctive and artisanal products.

Deep Learning

Introduction

A deep learning technology is based on artificial neural networks (ANNs). These ANNs constantly receive learning algorithms and continuously growing amounts of data to increase the efficiency of training processes. The larger the data volumes are, the more efficient this process is. The training process is called deep, because, with time passing, a neural network coversa growing number of levels. The deeper this network penetrates, the higher its productivity is. DL algorithms can create new tasks to solve current ones. One of the main benefits of deep learning over various machine learning algorithms is its ability to generate new features from a limited series of features located in the training dataset. Therefore, deep learning algorithms can create new tasks to solve current ones.

Applications of Deep Learning

- **Defence**
  The United States Department of Defence applied deep learning to train robots in new tasksthrough observation.

- **Bioinformatics**
  An auto encoder ANN was used in bioinformatics, to predict gene ontology annotations and gene-function relationships. In medical informatics, deep learning was used to predict sleep quality based on data from wearable's and predictions of health complications from electronic health record data. Deep learning has also shown efficacy in healthcare.

- **Self-Driving Cars**
  Deep Learning is the force that is bringing autonomous driving to life. A million sets of data are fed to a system to build a model, to train the machines to learn, and then test the results in asafe environment. Data from cameras,
  sensors, geo-mapping is helping create succinct and sophisticated models to navigate through traffic, identify paths.

CONVOLUTIONAL NEURAL NETWORKS (CNN)

A Convolutional Neural Network (CNN) is a type of neural network commonly used in deep learning for image and video recognition and processing tasks. It is inspired by the organizationof the visual cortex in animals, where neurons in the cortex are arranged in layers that respondto visual stimuli in a hierarchical manner.

CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers are responsible for feature extraction, where a filteris convolved with the input image to produce a feature map. The pooling layers down sample the feature maps by taking the maximum or average of each local region. The fully connected layers are responsible for classification or regression.
CNNs have shown great success in various computer vision tasks, including object recognition, image segmentation, and image captioning. They have also been applied to other domains such as natural language processing, speech recognition, and drug discovery.

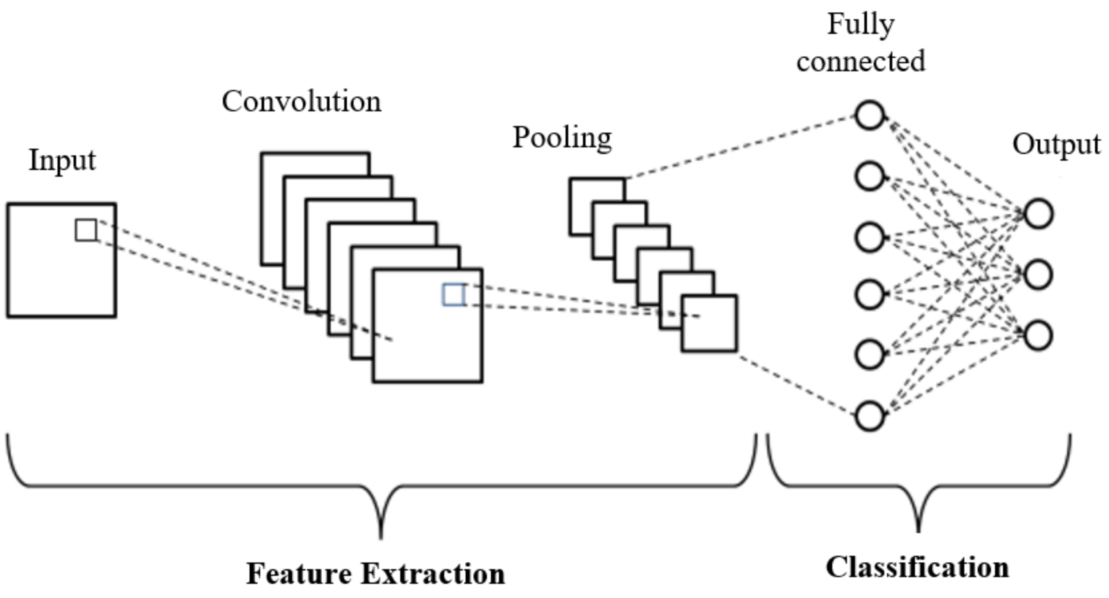
CNN layers

A deep learning CNN consists of three layers: a convolutional layer, a pooling layer and a fullyconnected (FC) layer. The convolutional layer is the first layer while the FC layer is the last. From the convolutional layer to the FC layer, the complexity of the CNN increases this increasing complexity that allows the CNN to successively identify larger portions and more complex features of an image until it finally identifies the object in its entirety.

Convolutional layer

The majority of computations happen in the convolutional layer, which is the core building block of a CNN. A second convolutional layer can follow the initial convolutional layer. The process of convolution involves a kernel or filter inside this layer moving across the receptive fields of the image, checking if a feature is present in the image.

Over multiple iterations, the kernel sweeps over the entire image. After each a dot product is calculated between the input pixels and the filter. The final output from the series of dots is known as a feature map or convolved feature. Ultimately, the image is converted into numericalvalues in this layer, which allows the CNN to interpret the image and extract relevant patterns from it.



Architecture of Convolutional Neural Networks

Pooling layer:

Like the convolutional layer, the pooling layer also sweeps a kernel or filter across the input image. But unlike the convolutional layer, the pooling layer reduces the numberof parameters in the input and also results in some information loss. On the positive side, this layer reduces complexity and improves the efficiency of the CNN.

Fully connected layer:

The FC layer is where image classification happens in the CNN basedon the features extracted in the previous layers. Here, fully connected means that all the inputsor nodes from one layer are connected to every activation unit or node of the next layer.

PRETRAINED MODELS

- **CNN:** "CNN" stands for Convolutional Neural Network. CNNs are a type of artificial neural network that have proven to be particularly effective in analysing visual imagery. They are widely used in tasks such as image recognition, object detection, and classification.

- **Mobile Net:** MobileNet architecture is based on depth-wise separable convolutions, which are composed of two main components: depth-wise convolutions and point-wise convolutions. This architecture significantly reduces the computational cost and number of parameters compared to traditional CNN architectures while maintaining reasonable accuracy.

- **VGG-16:** VGG16 stands for Visual Geometry Group 16-layer model. It consists of 16 convolutional and fully connected layers, with a simple and uniform architecture characterized by small 3x3 convolutional filters and max-pooling layers.

- **Inception V3:** This is designed to balance computational efficiency with improved performance compared to its predecessors. It features complex network architecture with deep convolutional layers and sophisticated modules called "Inception modules."

- **ResNET 50:** ResNet50 is based on the ResNet architecture, which introduced the concept of residual learning. Residual learning involves using shortcut connections (skip connections) to add the input of a layer to the output of a later layer, allowing the network to learn residual functions. ResNet50 specifically has 50 convolutional layers, including convolutional, pooling, and fully connected layers.

PROPOSED METHODOLOGY

Introduction

In today's textile industry, distinguishing between machine-made fabric and handmade fabric is crucial for various applications such as quality control, product authentication, and market segmentation. While traditional methods rely on manual inspection or expert judgment, advancements in deep learning offer a promising avenue for automated classification based on visual characteristics.

This proposed methodology outlines a systematic approach to differentiate between machine-made fabric and handmade fabric using Convolutional Neural Networks (CNNs). CNNs are a class of deep neural networks specifically designed for image analysis tasks, making them well-suited for this project's objective of fabric classification based on visual features.

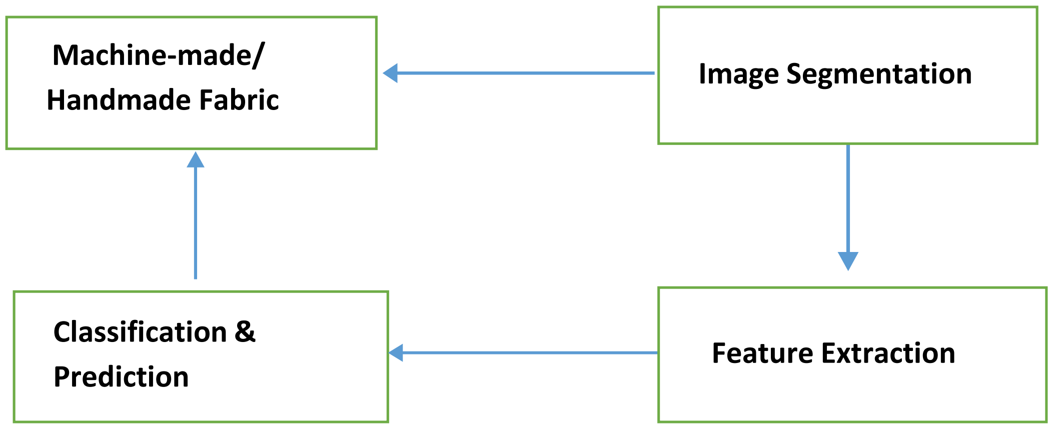
The methodology begins with data collection, where a diverse dataset of labelled images representing both machine-made and handmade fabric is assembled. These images undergo pre-processing to ensure uniformity in size and pixel values, followed by splitting into training, validation, and test sets.

Next, CNN architecture is designed to learn and extract discriminative features from the fabric images. Various architectures and hyper-parameters are explored and optimized to achieve the best classification performance.

The model is then trained using the training set while monitoring its performance on the validation set to prevent over fitting. Evaluation metrics such as accuracy, precision, recall, and F1-score are computed to assess the model's effectiveness in differentiating between fabric types.

Based on insights gained from the evaluation phase, the model may undergo fine-tuning to further improve its performance. Once deemed satisfactory, the trained model is deployed to classify new fabric images as either machine-made or handmade.

Block Diagram



Proposed System

1. Texture pattern

The texture patterns of machine-made fabric and handmade fabric can vary due to differences in production methods, materials used, and the human touch involved in the creation process.

Handmade Fabric

Machine-made Fabric

2. Thread Pattern

The threads used in handmade fabric and machine-made fabric differ in various aspects due to the distinct production methods employed. Threads used in machine-made fabric are typically more consistent in thickness, texture, and colour compared to handmade fabric threads. Machine-made fabric threads are often stronger and more durable due to the controlled and standardized manufacturing process.

3. **Weaving Technique**

Handmade fabric can be woven using various techniques, each of which imparts unique characteristics to the final product. Some common weaving techniques used in handmade fabric production include:

- Plain Weave
- Twill Weave
- Satin Weave
- Basket Weave

Data Collection

Determine the specific types of machine-made and handmade fabrics you want to include in your dataset. Decide on the number of images you need for each fabric type and any specific characteristics or variations you want to capture. Look for online databases, repositories, or websites that provide high-quality images of fabric samples. Consider reaching out to textile manufacturers, artisans, or fabric suppliers who may be willing to provide images of their products. If you can't find suitable images from existing sources, consider web scraping techniques to collect images from online sources such as e-commerce websites, blogs, or social media platforms. Ensure that you comply with copyright laws and website terms of use when scraping images from online sources. In addition to automated scraping, manually collect images from various sources such as online galleries, forums, or personal collections. Include images that showcase different fabric textures, patterns, colours, and variations to create a diverse dataset.

Sample Data

Type of Fabric	No. of training samples	No. of Validation Samples	No. of testing samples
Machine-made	1560	195	195
Handmade	1560	195	195

Total No. of Samples=3900

Data Pre-processing

Data pre-processing is a crucial step in preparing your fabric image dataset for training a Convolutional Neural Network (CNN) in your project.

- Resizing
- Normalization
- Data Augmentation
- Train-Validation-Test Split
- Data Quality Check

The VGG16 model, or Visual Geometry Group 16, is a highly effective Convolutional Neural Network (CNN) primarily designed for image classification tasks. Developed by the Visual Geometry Group at the University of Oxford for the 2014 ImageNet Large Scale Visual Recognition Challenge, VGG16 features a simple yet powerful architecture with 16 layers, including 13 convolutional layers followed by Rectified Linear Unit (ReLU) activation functions. The model employs small receptive fields (3x3) with a stride of 1, ensuring the capture of intricate patterns.

Max pooling layers facilitate down-sampling, optimizing computational efficiency. Fully connected layers generate predictions based on learned features, making VGG16 particularly suitable for image classification. Training involves adjusting parameters using labelled datasets, with transfer learning being a common practice for fine-tuning on specific tasks. VGG16's applications extend beyond image classification to object detection, image segmentation, and artistic style transfer, showcasing its versatility in various image-related domains.

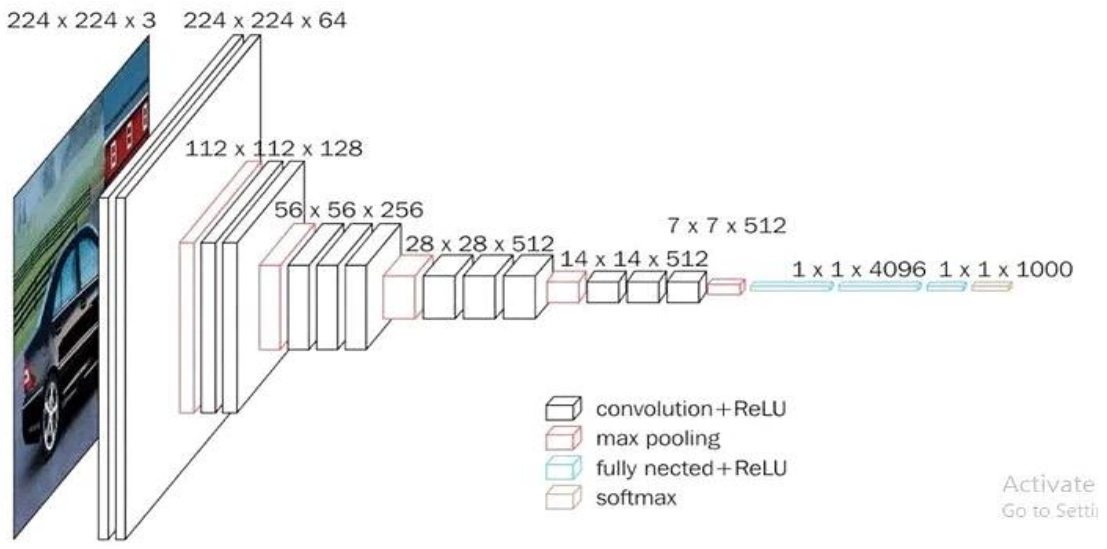
VGG 16 Architecture

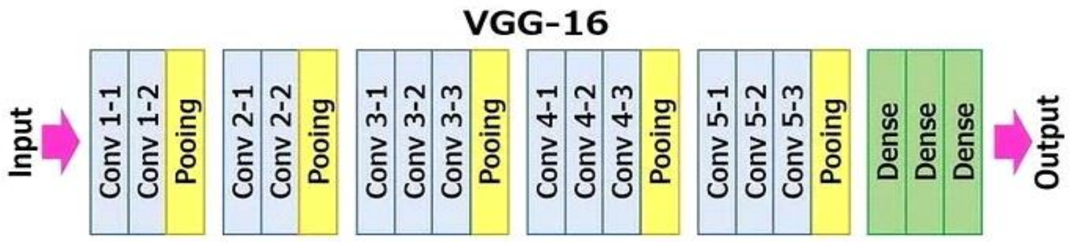
- The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer.

- VGG16 takes input tensor size as 224, 244 with 3 RGB channel

- Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2.

- The convolution and max pool layers are consistently arranged throughout the whole architecture
- Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters.
- Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.





VGG-16 Architecture

CLASSIFICATION

Classification is a technique where we categorize data into a given number of classes. The main goal of a classification problem is to identify the category/class to which a new data willfall under. Image Classification is the task of assigning an input image one label from a fixed set of categories. This is one of the core problems in Computer Vision that, despite its simplicity, has a large variety of practical applications. Many other seemingly distinct Computer Vision tasks (such as object detection, segmentation) can be reduced to imageclassification.

Terminology

- Classifier: An algorithm that maps the input data to a specific category.
- Classification model: A Classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data.
- Feature: A feature is an individual measurable property of a phenomenon being observed.
- Classification: Classification task with five different classes with one possible outcomeof classified and quality.
- Multi class classification: Classification with more than two classes. In multi class classification each sample is assigned to one and only one target label. E.g., An animalcan be cat or dog but not both at the same time.
- Multi label classification: Classification task where each sample is mapped to a set oftarget labels (more than one class). E.g., A news article can be about sports, a person, and location at the same time.

Convolutional

Convolution is a process which is used for tasks such as filtering, edge detection, and picture sharpening on images. Different convolution kernels are used to achieve this. A neural network may utilize convolution to extract information from an image.

$$c(x) = \int_{-\infty}^{\infty} f(\tau)\, g(x-\tau)\, d\tau$$

A function c(x) is generated with the integral of the product of the overlapping function valuesof the functions f and g. These will be flipped and altered for better result.

Pooling Layer

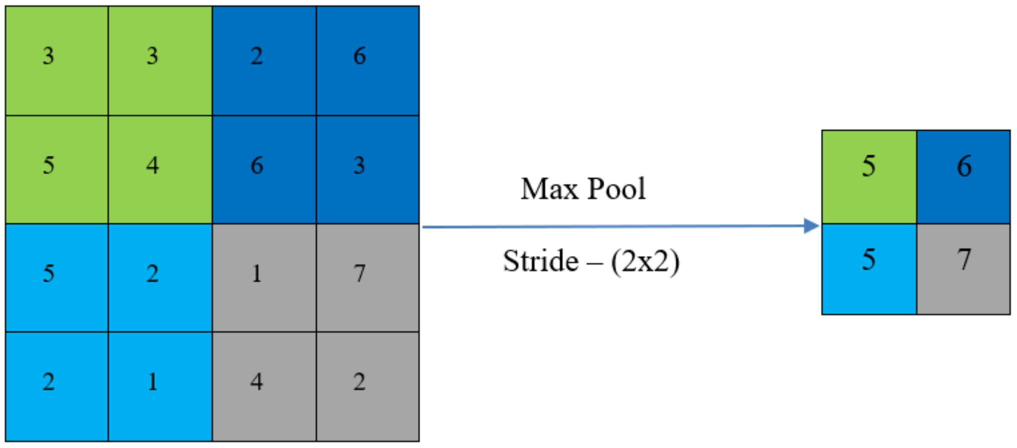
Pooling is the technique implemented in order to reduce the dimensions of the feature map andmake it easy for computations. Pooling will preserve the data meaning and reduce the size of the content as well. Pooling consists of an n x n size filter and a stride representing the step size.
Consider H*W*C-sized feature map the result of pooling a layer is

$$(H-Fs+1)/st*(W-f+1)/st*C$$

where, H - height of feature tree, W - width of feature tree, C - number of channels in the featuremap, Fs - size of filter, st - stride length.

Max Pooling

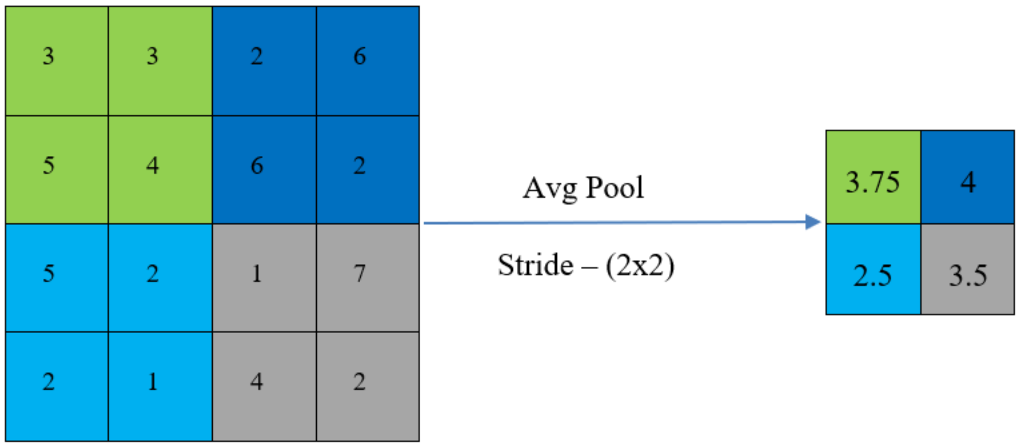
Max pooling is done in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parametersto learn and provides basic translation invariance to the internal representation. Max pooling isdone by applying a max filter to (usually) non-overlapping sub regions of the initial representation.



Max Pooling

Average Pooling

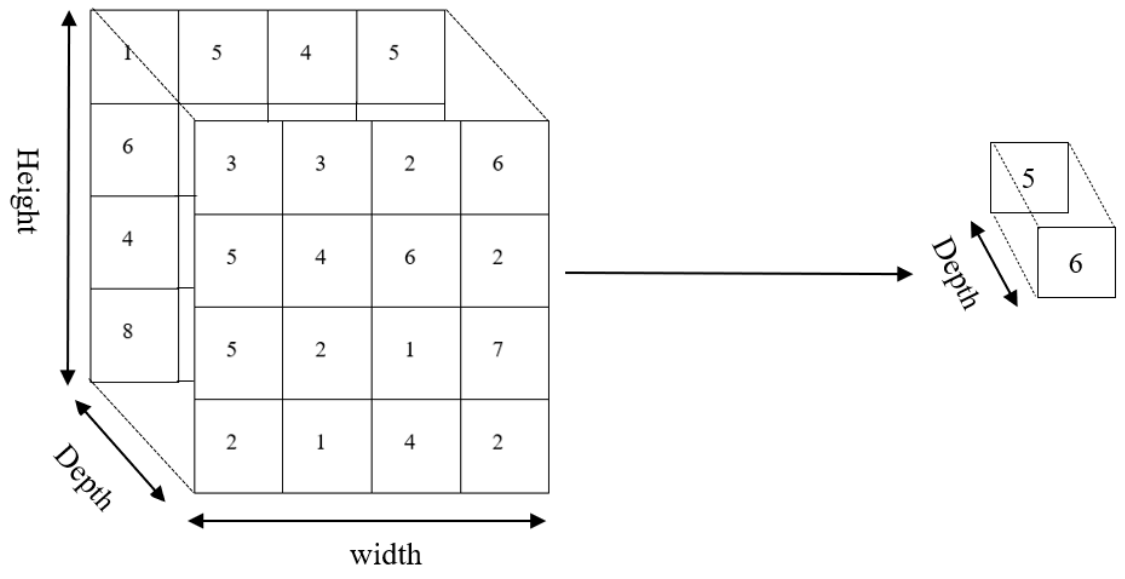
This is a pooling operation that calculates the average value for patches of a featuremap, and uses it to create a down sampled (pooled) feature map. It is usually used after a convolutional layer. It adds a small amount of translation invariance – meaning translating the



Average Pooling

Global Average Pooling

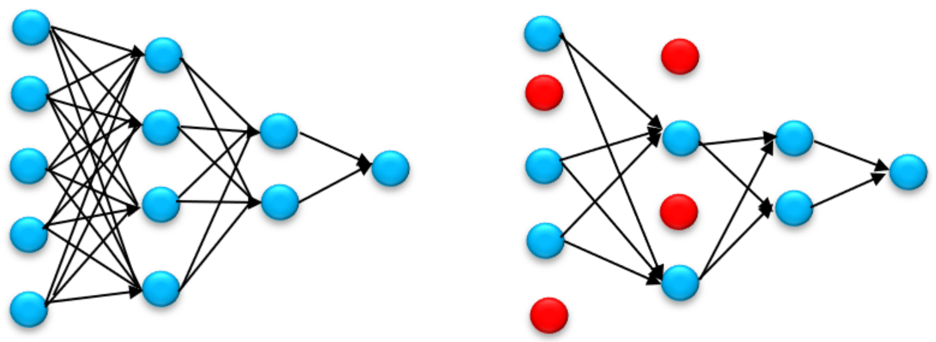
This is a pooling operation designed to replace fully connected layers in classical CNNs. The idea is to generate one feature map for each corresponding category of theclassification task in the last mlpconv layer. Instead of adding fully connected layers on top ofthe feature maps, we take the average of each feature map, and the resulting vector is fed directlyinto the soft max layer.
One advantage of global average pooling over the fully connected layers is that it is more native to the convolution structure by enforcing correspondences between feature maps and categories. Thus, the feature maps can be easily interpreted as categories of confidence maps. Another advantage is that there is no parameter to optimize in the global average pooling thus overfittingis avoided at this layer. Furthermore, global average pooling sums out the spatial information, thus it is more robust to spatial translations of the input.
The 2D Global average pooling block takes a tensor of size (input width) x (input height) x (input channels) and computes the average value of all values across the entire (input width) x(input height) matrix for each of the (input channels). The output is thus a 1- dimensional tensorof size (input channels).



Global Average Pooling 2D

Dropout

Dropout Layer is one of the most popular regularization techniques to reduce over fitting in thedeep learning models. Over fitting in the model occurs when it shows more accuracy on the training data but less accuracy on the test data or unseen data.
In the dropout technique, some of the neurons in hidden or visible layers are dropped or omitted randomly. The experiments show that this dropout technique regularizes the neural network model to produce a robust model which does not over fit. Problem with Over fitting is, large neural nets trained on relatively small datasets can over fit the training data. This has the effectof the model learning the statistical noise in the training data, which results in poor performancewhen the model is evaluated on new data, e.g., a test dataset. Generalization error increases dueto over fitting.



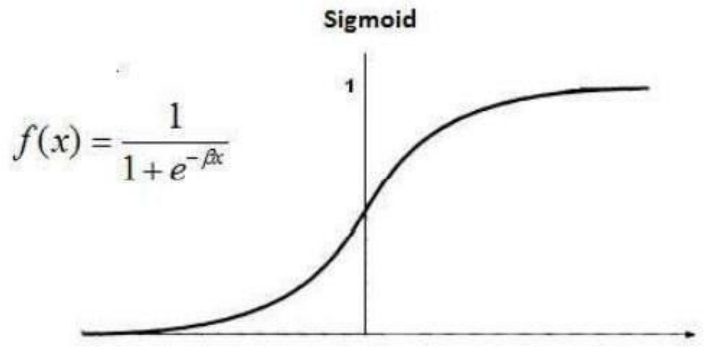
(a) Standard Neural Network (b) After applying Dropout

ACTIVATION FUNCTIONS

Sigmoid

The sigmoid transformation generates a smoother range of values between 0 and 1. We might need to observe the changes in the output with slight changes in the input values. Smooth curvesallow us to do that and are hence preferred over step functions.

$$\mathbf{Sigmoid(x) = 1/(1+e^{-x})}$$

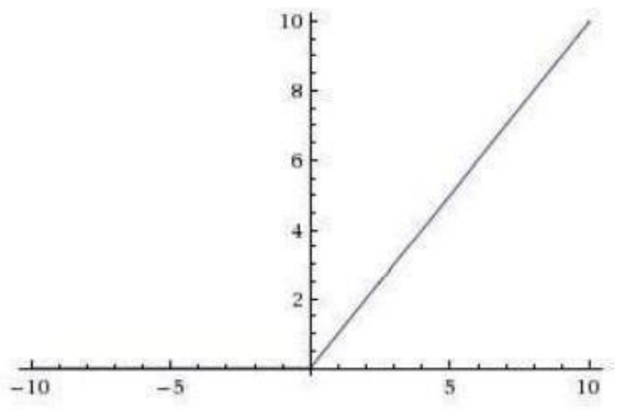


Sigmoid Function

RELU (Rectified Linear Unit):

Instead of sigmoid, the recent networks prefer using ReLu activation functions for the hidden layers. The function is defined as:

F(x) = max (x, 0)



RELU Activation Function

The major benefit of using ReLU is that it has a constant derivative value for all inputs greaterthan 0. ReLU is linear (identity) for all positive values, and zero for all negative values. This means that:

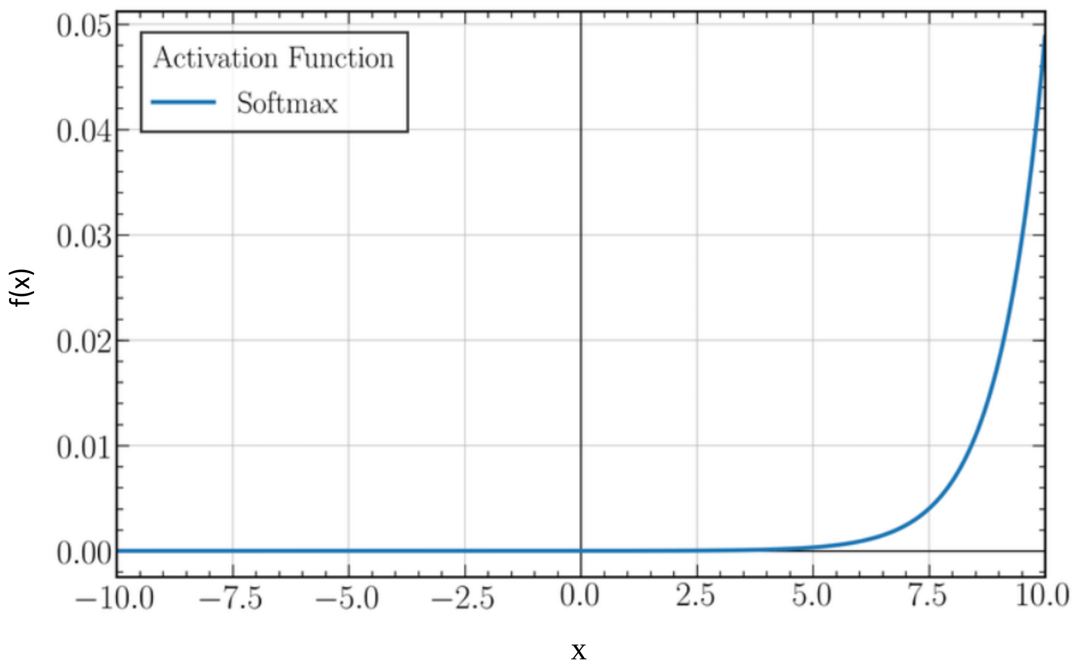
● It’s cheap to compute as there is no complicated math. The model can therefore take less time to train or run.

- It converges faster. Linearity means that the slope doesn't plateau, or "saturate," when x getslarge. It doesn't have the vanishing gradient problem suffered by other activation functionslike sigmoid or tanh.
- It's sparsely activated. Since ReLU is zero for all negative inputs, it's likely for any given unit to not activate at all. This is often desirable.

Softmax

Softmax activation functions are normally used in the output layer for classification problems. It is similar to the sigmoid function, with the only difference being that the outputs are normalized to sum up to 1.

$$S\ (z_i) = \frac{\exp\ (z_i)}{\sum \exp\ (z_j)_{j=1}^{k}}$$



Softmax Activation Function

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

Processor	:	Intel i5 and above
Ram	:	8 GB and above
Hard Disk	:	1 TB and above

SOFTWARE REQUIREMENTS

Programming language	:	Python 3.7
IDE	:	Jupyter Notebook
Cloud Service	:	AWS

LIBRARIES USED

PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by GuidoVan Rossum and first released in 1991, Python has A Design Philosophy That Emphasizes code readability, notably using significant whitespace. It has a wide range of applications from Web development (like: Django and Bottle), Scientific and mathematical computing (Orange, SciPy, NumPy) to desktop Graphical User Interfaces (Pygame, Panda3D).

KERAS

Kerasisan Open-Source Neural Network library written Python that runs on top of Theanoor TensorFlow. It is designed to be modular, fast and easy to use. It was developed by François Cholet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the

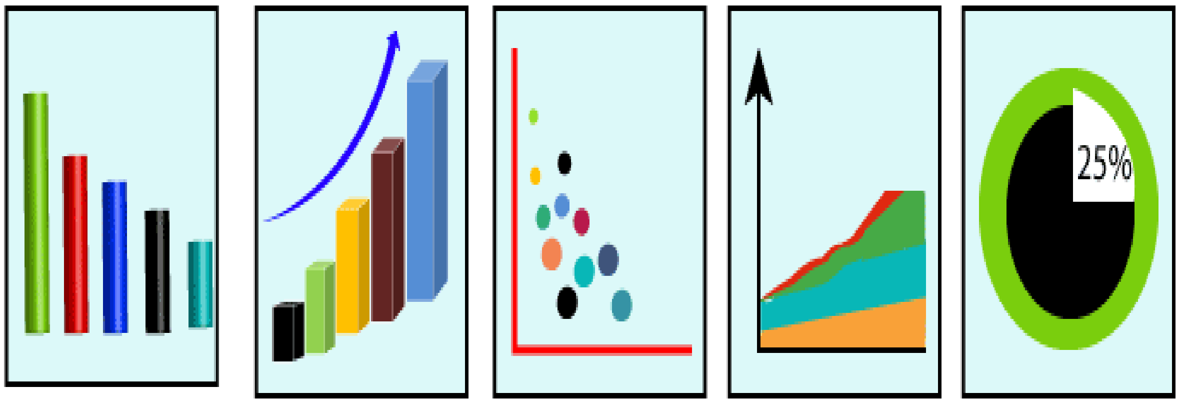
"Backend. So Keras is a high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano. Keras Doesn't handle Low-Level API such as making the computational graph, making tensors or other variables becauseit has been handled by the "backend" engine. Keras High-Level API.

TENSORFLOW

Tensor Flow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. It is in the industry to have experience.

MATPLOTLIB

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with thebroader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

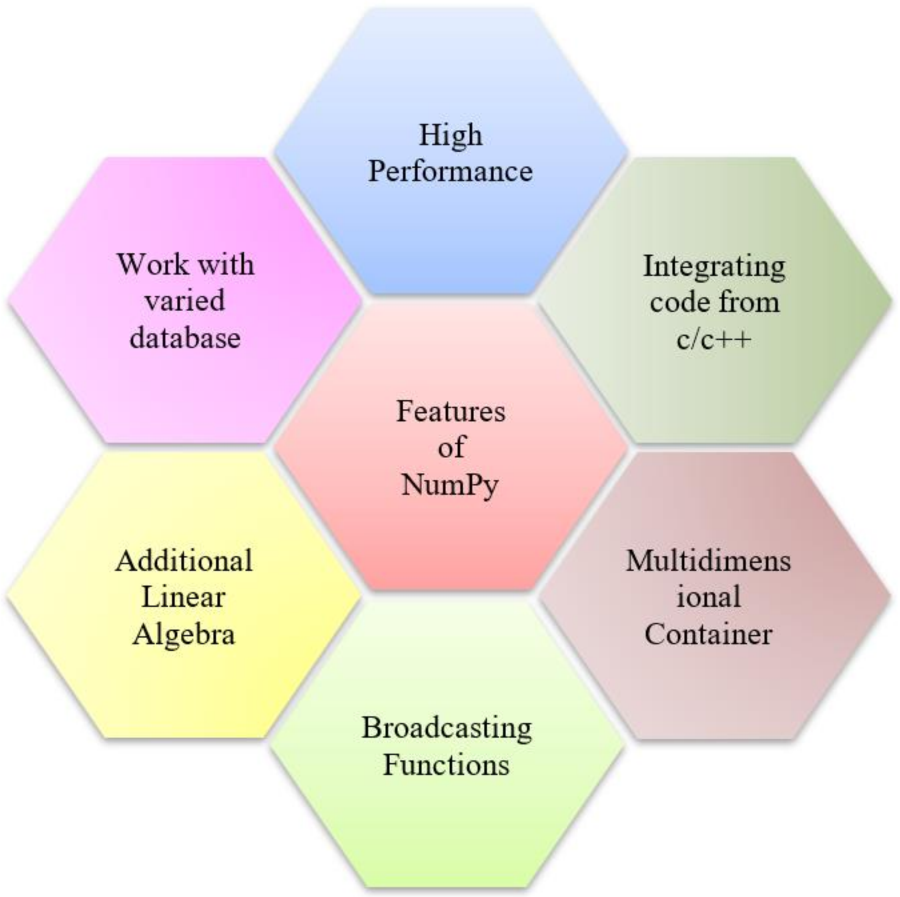


(a) Bar Graph (b) Histogram (c) Scatter Plot (d) Area Plot (e) Pie Plot

Key Plots for Data Visualization

NUMPY

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.


High Performance
Work with varied database
Integrating code from c/c++
Features of NumPy
Additional Linear Algebra
Multidimensional Container
Broadcasting Functions

Features of Numpy

SYSTEM DESIGN

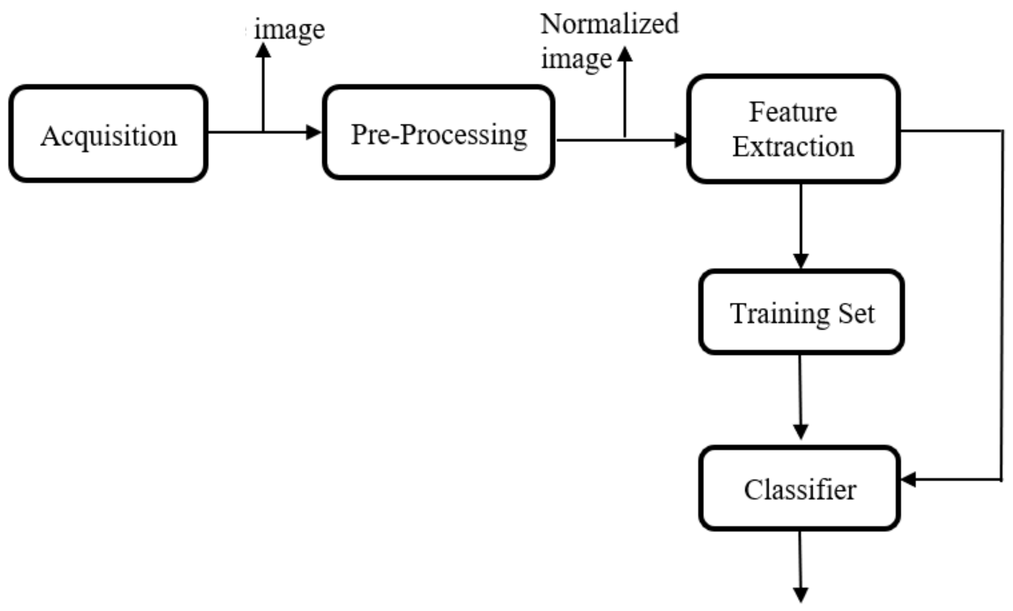
DATA FLOW DIAGRAM

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be usedto represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

2. The Data Flow Diagram (DFD) is one the most important modeling tools. opts used modelthe system components. These components are the system process, the data used by the process,an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flown the transformations that are applied as data moves from input to output.

4. DFD is also known as bubble chart. A DFD may be used to represent a system at any levelof abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



The process of training

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 annotation. In the future, some form of method or process may also be added to; orassociated 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.

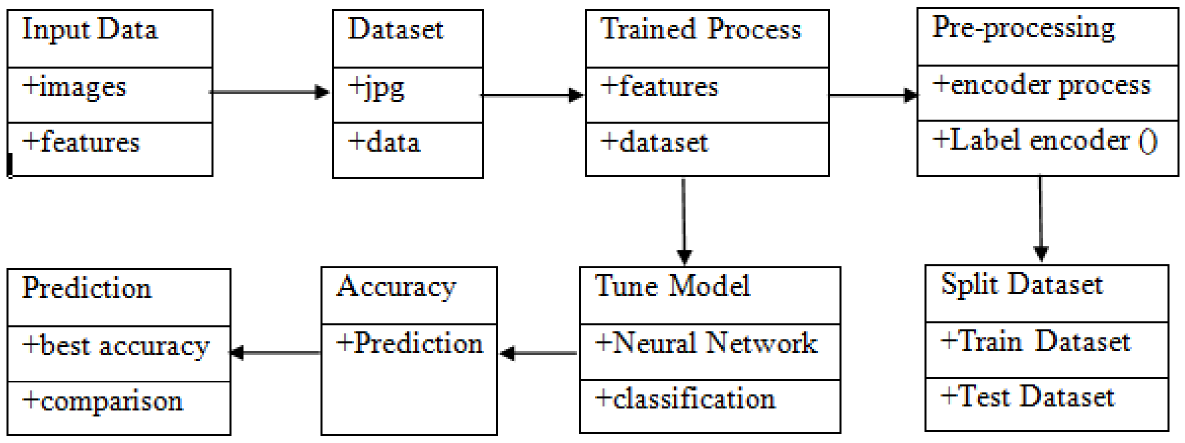
The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type ofstatic 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.

Class diagram describes the attributes and operations of a class and also the constraints imposedon the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

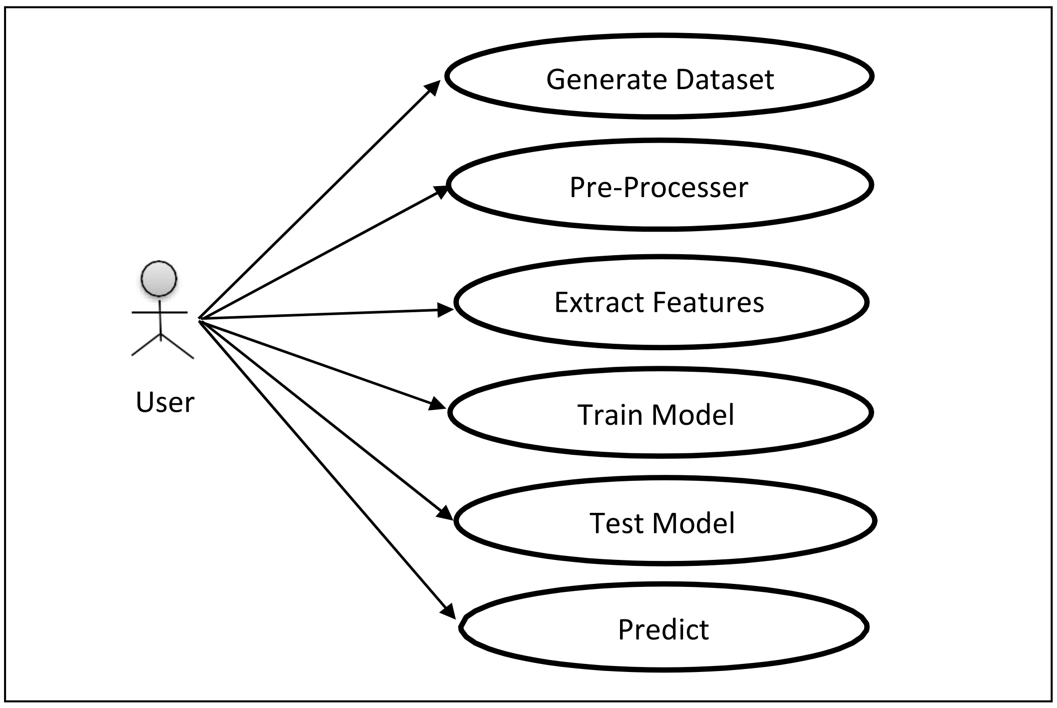
Class diagram shows a collection of classes, interfaces, associations, collaborations, andconstraints. It is also known as a structural diagram.



Class Diagram

USE CASE DIAGRAM

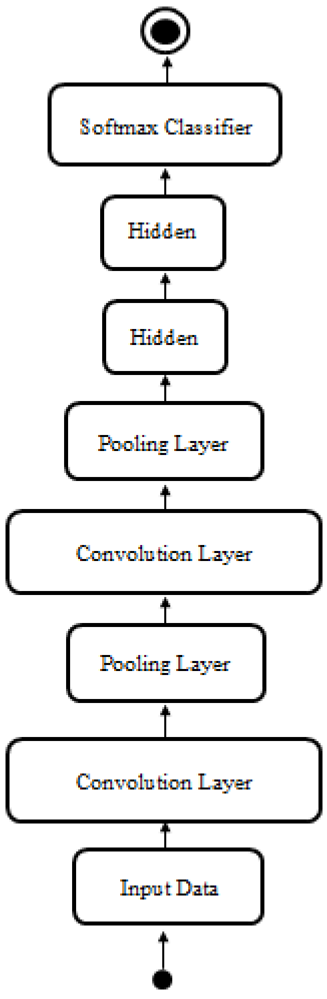
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagramdefined by and created from a Use-case analysis. Its purpose is to present a graphical overviewof the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies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



Use Case Diagram

STATE CHART DIAGRAM

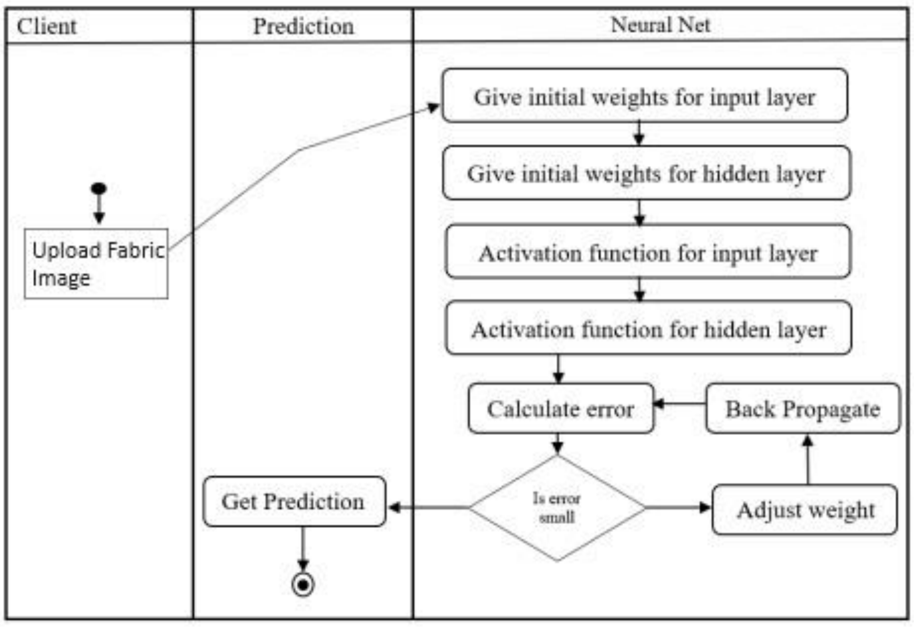
State diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable.



State Chart Diagram

ACTIVITY DIAGRAM

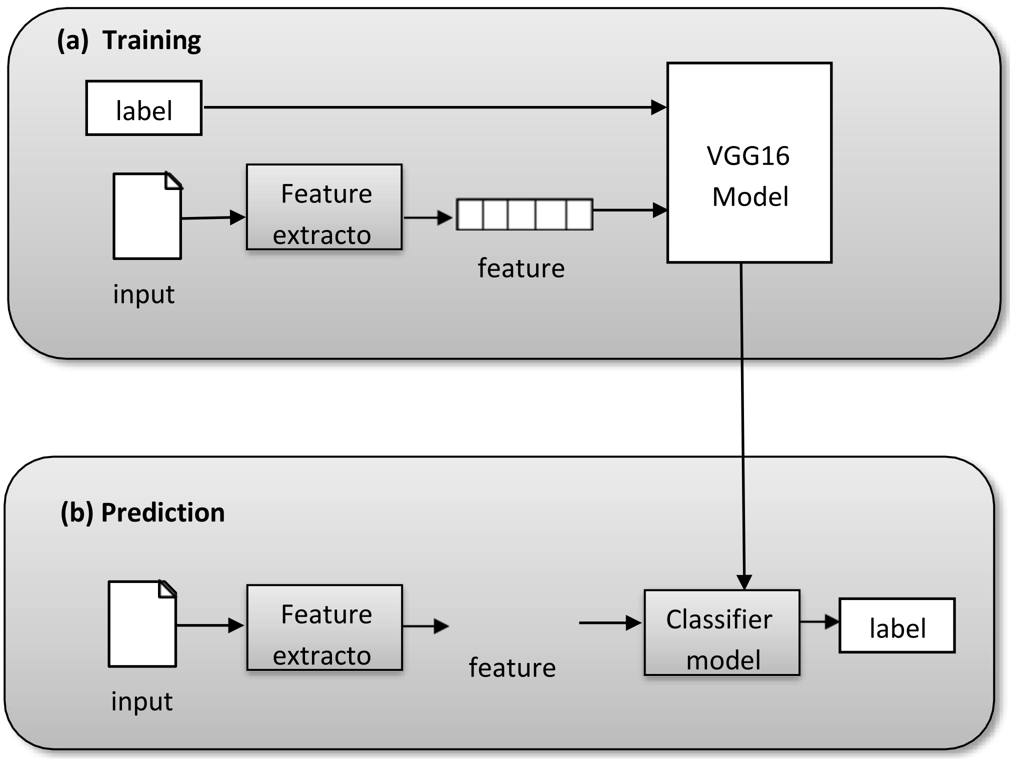
Activity diagram is another important diagram in UML to describe the dynamic aspects of thesystem. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.


Client
Prediction
Neural Net
Upload Fabric Image
Give initial weights for input layer
Give initial weights for hidden layer
Activation function for input layer
Activation function for hidden layer
Calculate error
Back Propagate
Is error small
Adjust weight
Get Prediction

Activity Diagram

OBJECT DIAGRAM

Object diagrams are derived from class diagrams so object diagrams are dependent upon classdiagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.
Object diagrams are used to render a set of objects and their relationships as an instance. The purposes of object diagrams are similar to class diagrams. The difference is that a class diagram represents an abstract model consisting of classes and their relationships. However, an object diagram represents an instance at a particular moment, which is concrete in nature.



Object Diagram

STREAMLIT

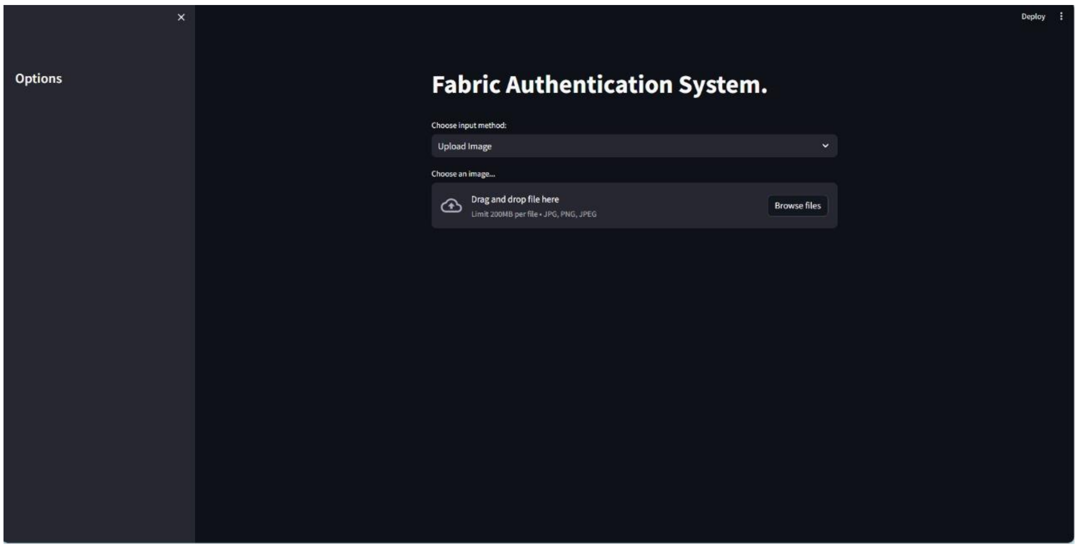
Streamlit is an open-source Python framework used for building web applications focused on data science, machine learning, and other interactive data-related tasks. It allows developers to create web applications quickly and easily using familiar Python scripting. Streamlit is a powerful and user-friendly framework for building interactive web applications for data science, machine learning, and data analysis projects, offering a balance between ease of use and flexibility for developers.

USES

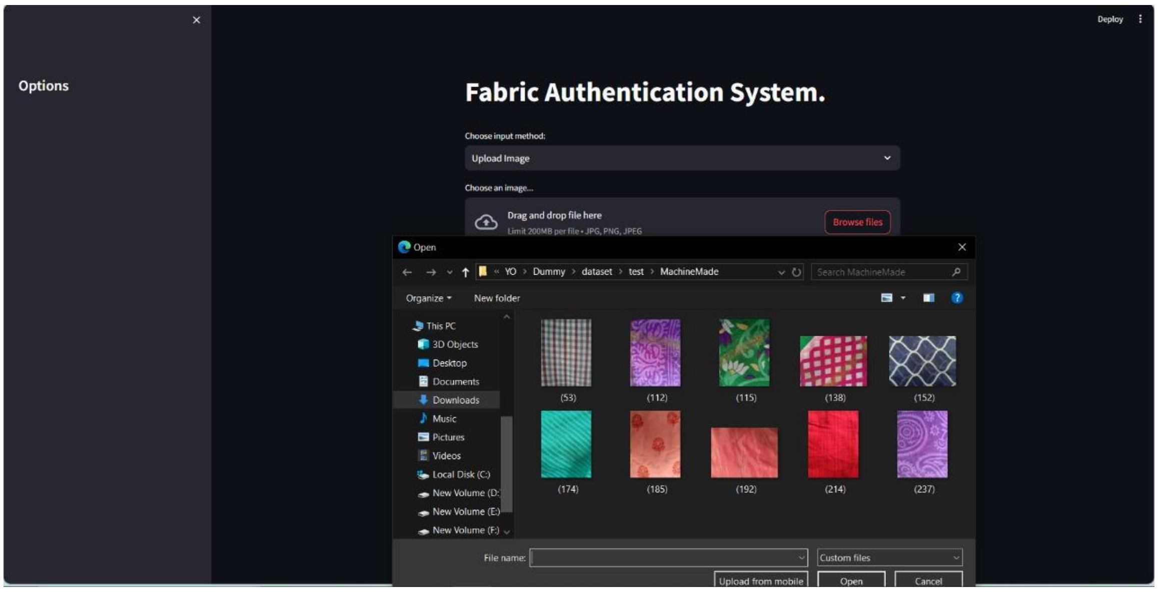
- Rapid Development
- Data Visualization
- Interactive Widgets
- Machine Learning and Data Analysis
- Sharing and Deployment

Streamlit Interface

- **GUI of Streamlit**



- **Uploading Image**



- **Successfully Image uploading**

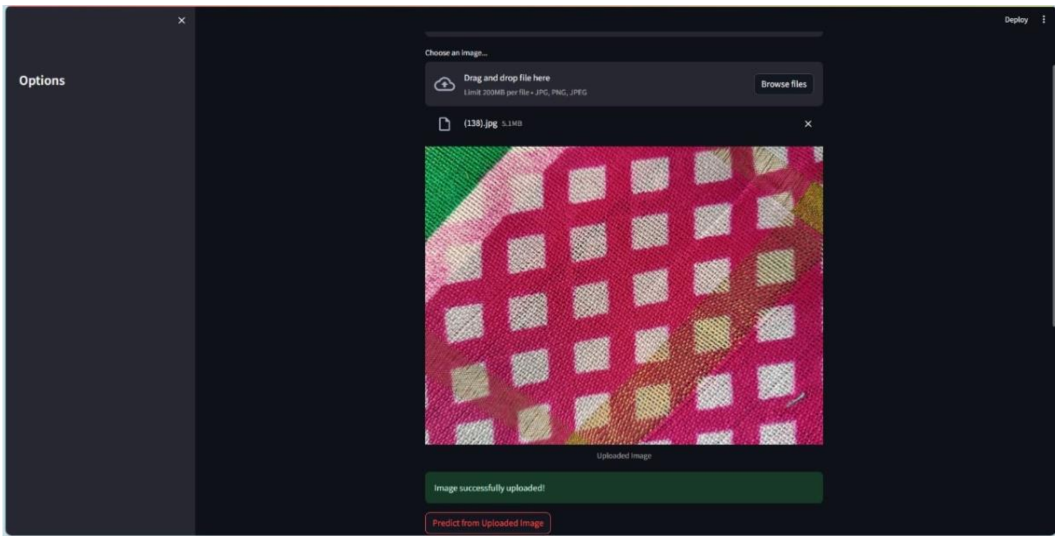


Image successfully uploaded!

- **Fabric Prediction**

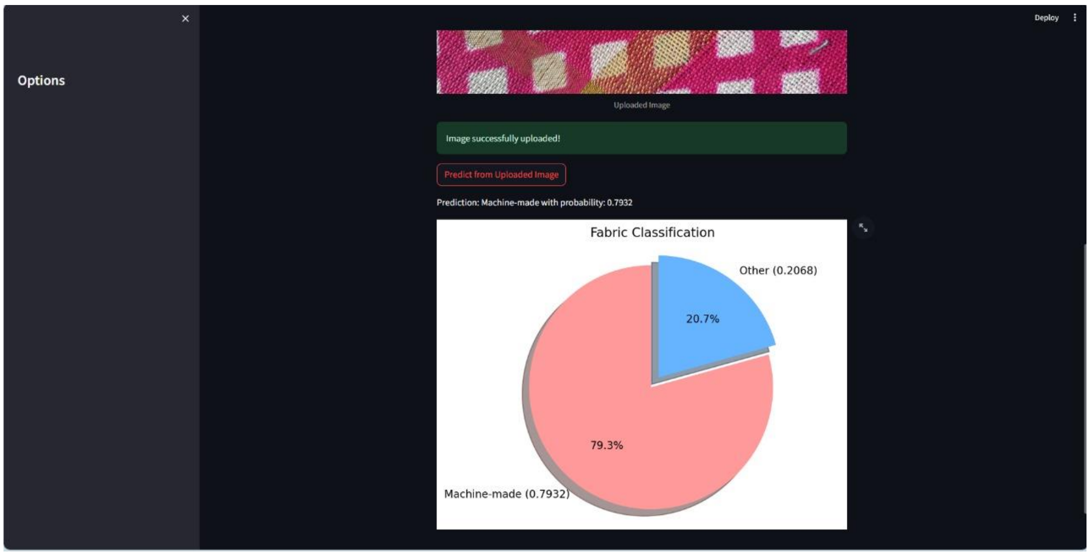


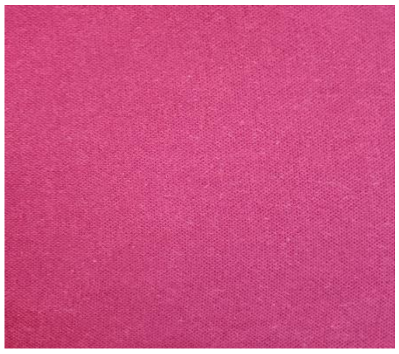
Image successfully uploaded!

Prediction: Machine-made with probability: 0.7932

RESULTS

IMAGE DATASET:

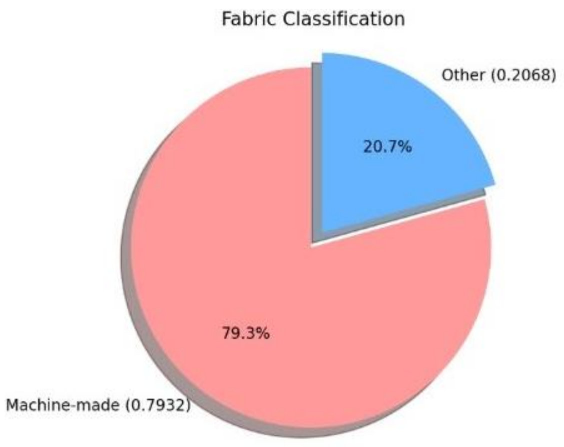
The following are some real-time images we have collected to train and test the system.


Machine-made Fabric

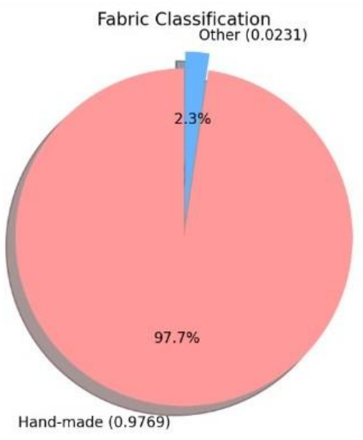
Handmade Fabric

FINAL OUTPUT:

The following are the results of above Images.



Machine-made Prediction



Handmade Prediction

Prediction of fabric

Model Deploying (AWS)

Introduction of AWS

Amazon Web Services (AWS) is a comprehensive and widely-used cloud computing platform provided by Amazon.com. It offers a wide range of cloud services, including computing power, storage solutions, networking, databases, machine learning, artificial intelligence, Internet of Things (IoT), security, and more. AWS provides these services on a pay-as-you-go basis, allowing businesses and developers to scale resources up or down as needed without the need for large upfront investments in infrastructure.

AWS Services

- **Compute:** Services like Amazon Elastic Compute Cloud (EC2) provide resizable compute capacity in the cloud, allowing users to run virtual servers for various workloads.

- **Storage**: AWS offers scalable storage solutions such as Amazon Simple Storage Service (S3) for object storage, Amazon Elastic Block Store (EBS) for block storage, and Amazon Glacier for long-term archival.

- **Database**: AWS provides a range of managed database services, including Amazon Relational Database Service (RDS) for relational databases, Amazon Dynamo DB for NoSQL databases, and Amazon Redshift for data warehousing.

- **Networking**: AWS offers services for networking and content delivery, such as Amazon Virtual Private Cloud (VPC) for creating isolated virtual networks, Amazon Route 53 for domain name system (DNS) management, and Amazon Cloud Front for content delivery.

- **Security and Identity**: AWS provides security services including identity and access management (IAM), AWS Identity and Access Management, as well as encryption, key management, and compliance services.

- **Developer Tools**: AWS provides a range of developer tools, including AWS Code Pipeline for continuous integration and continuous delivery (CI/CD), AWS Code Build for building and testing code, and AWS Code Deploy for automating code deployments.

EC2 Instance:

Amazon Elastic Compute Cloud (Amazon EC2) is one of the core services provided by Amazon Web Services (AWS). It allows users to rent virtual servers, known as instances, to run their applications. EC2 provides scalable compute capacity in the cloud, allowing users to launch instances with a variety of operating systems, configurations, and pricing options.

Creation of EC2 Instance:

- **Sign in to AWS Management Console**: Log in to your AWS account using your credentials. Once logged in, you'll be taken to the AWS Management Console.
- **Navigate to EC2 Dashboard**: From the AWS Management Console, navigate to the EC2 service by either searching for "EC2" in the search bar or locating it under the "Compute" section.
- **Launch Instance**: Once in the EC2 dashboard, click on the "Launch Instance" button to begin the process of creating a new EC2 instance.
- **Choose an Amazon Machine Image (AMI)**: An AMI is a pre-configured template for the virtual machine, including the operating system and any additional software. AWS offers a wide range of AMIs to choose from, including various Linux distributions, Windows Server, and more. Select the AMI that best suits your requirements.
- **Choose an Instance Type**: Instance types determine the hardware resources (CPU, memory, storage, etc.) allocated to your instance. AWS offers a variety of instance types optimized for different use cases, such as general-purpose, compute-optimized, memory-optimized, etc. Choose the instance type that aligns with your workload's requirements and budget.

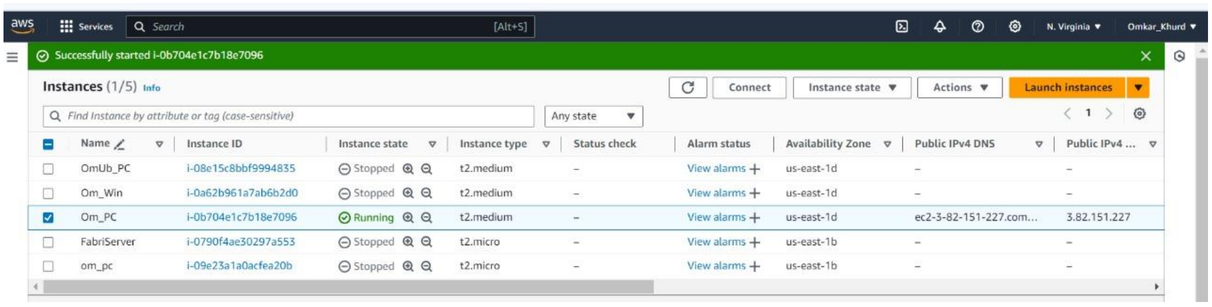
- **Configure Instance**: Configure the instance settings, including the number of instances to launch, network settings (VPC, subnet, security groups), storage (EBS volumes), and any additional configuration options.
- **Add Storage:** Specify the storage requirements for your instance. You can add Elastic Block Store (EBS) volumes to your instance to provide persistent block-level storage.
- **Configure Security Group**: Security groups act as virtual firewalls for your instance, controlling inbound and outbound traffic. Configure the security group rules to allow access to your instance based on your application's requirements (e.g., SSH access, HTTP access).
- **Review and Launch**: Review the configuration details of your instance to ensure everything is set up correctly. Once you're satisfied, click on the "Launch" button.
- **Create Key Pair**: If you don't already have a key pair, you'll be prompted to create one. A key pair is required to securely connect to your instance using SSH. Download the key pair file (.pem) and store it in a secure location.
- **Launch Instance**: After creating the key pair, click on the "Launch Instances" button to launch your EC2 instance.
- **Access your Instance**: Once your instance is launched, you can connect to it using SSH (for Linux instances) or Remote Desktop Protocol (RDP) (for Windows instances) using the key pair you created earlier.

AWS INTERFACE :

Step-1:

we are create EC2 instance. we create costume machine on AWS.

machine name is Amazon Linux machine.



Successfully started i-0b704e1c7b18e7096

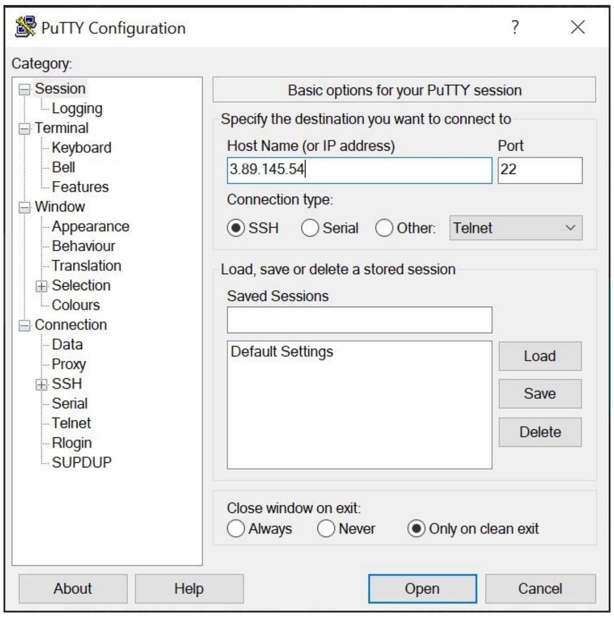
Instances (1/5) Info

Find Instance by attribute or tag (case-sensitive)

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
☐	OmUb_PC	i-08e15c8bbf9994835	Stopped	t2.medium	–	View alarms +	us-east-1d	–	–
☐	Om_Win	i-0a62b961a7ab6b2d0	Stopped	t2.medium	–	View alarms +	us-east-1d	–	–
☑	Om_PC	i-0b704e1c7b18e7096	Running	t2.medium	–	View alarms +	us-east-1d	ec2-3-82-151-227.com...	3.82.151.227
☐	FabriServer	i-0790f4ae30297a553	Stopped	t2.micro	–	View alarms +	us-east-1b	–	–
☐	om_pc	i-09e23a1a0acfea20b	Stopped	t2.micro	–	View alarms +	us-east-1b	–	–

Step 2:

for connecting server of ec2 instance, we need Putty application



Specify the destination you want to connect to

Host Name (or IP address): 3.89.145.54

Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

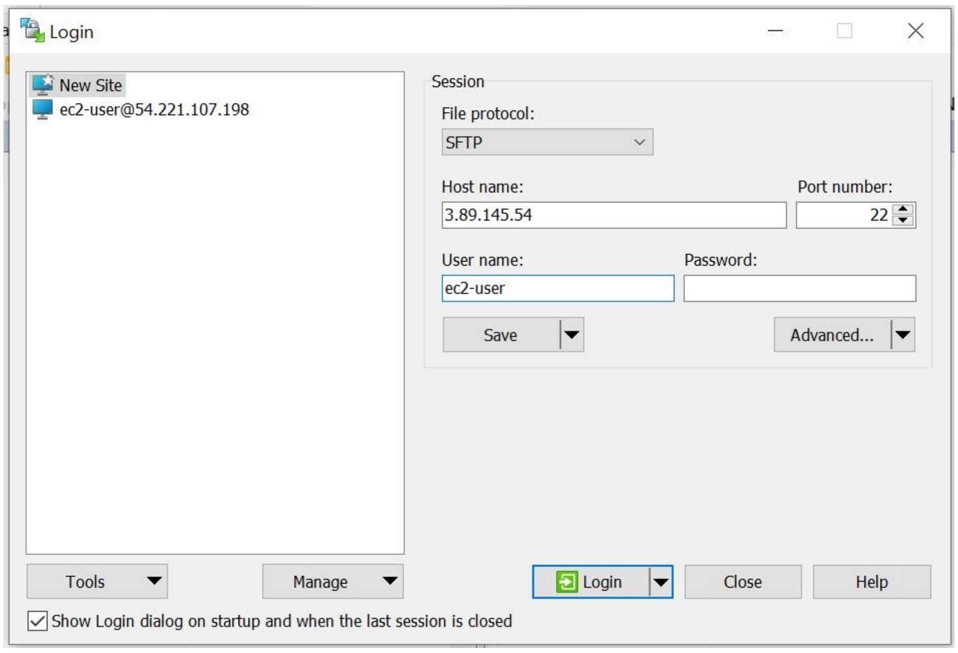
Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Step-3:

for sharing files to our ec2 instance directly, we need to use WinSCP application



Login

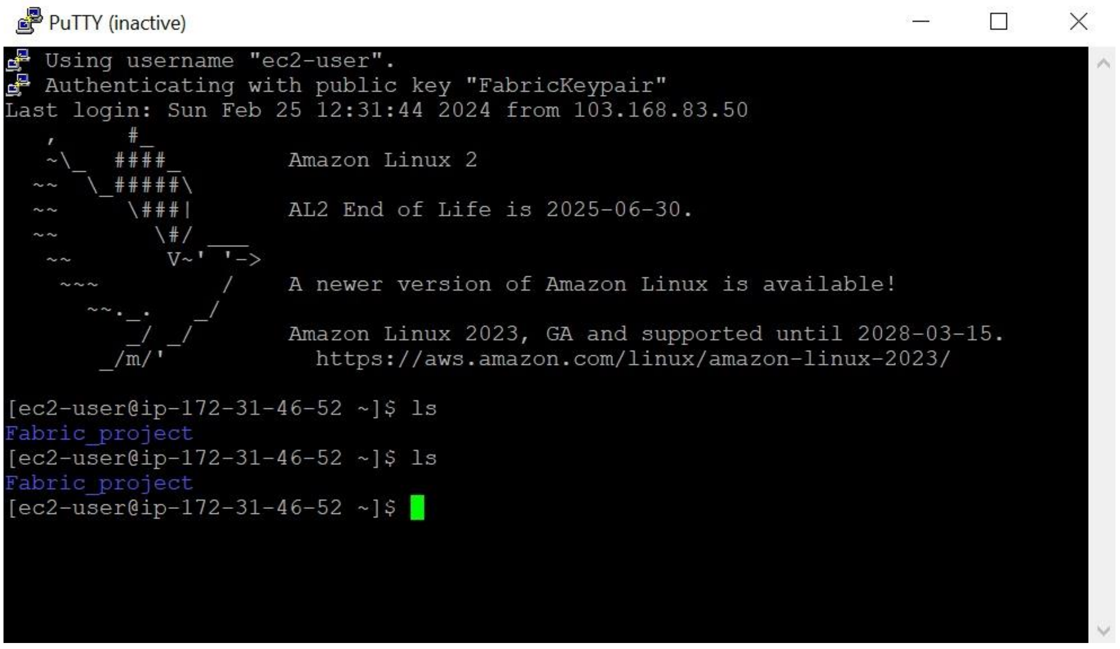
New Site
ec2-user@54.221.107.198

Session
File protocol: SFTP
Host name: 3.89.145.54
Port number: 22
User name: ec2-user
Password:

Save
Advanced...
Tools
Manage
Login
Close
Help
Show Login dialog on startup and when the last session is closed

Step-4:

CMD interface



PuTTY (inactive)

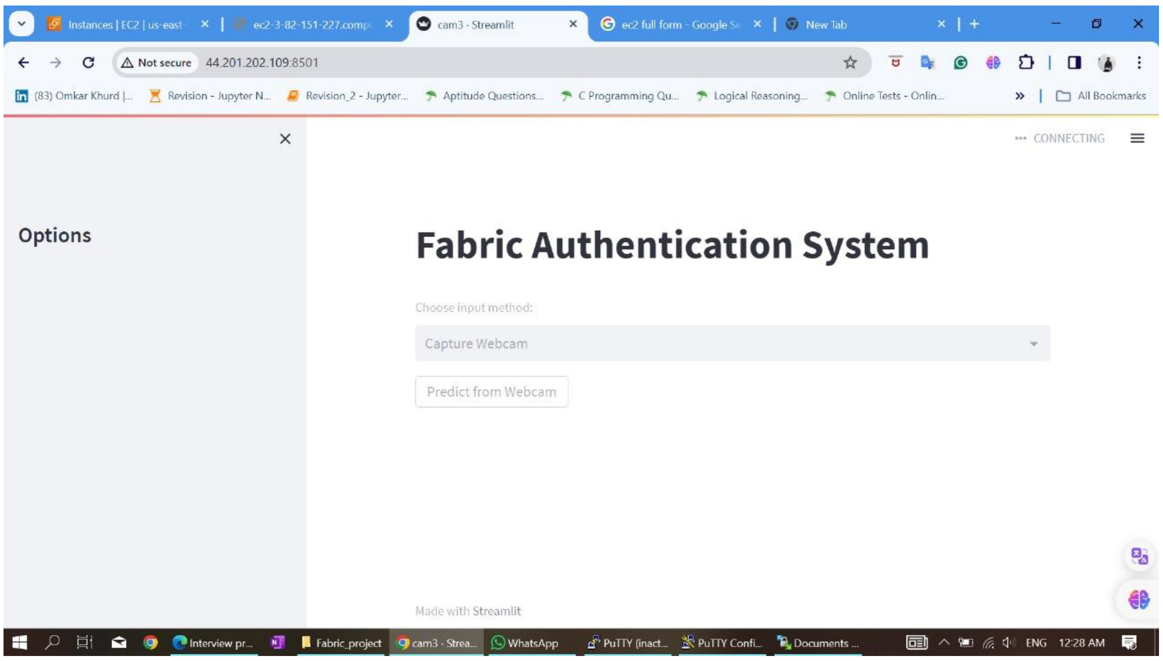
```
Using username "ec2-user".
Authenticating with public key "FabricKeypair"
Last login: Sun Feb 25 12:31:44 2024 from 103.168.83.50
    ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2025-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-46-52 ~]$ ls
Fabric_project
[ec2-user@ip-172-31-46-52 ~]$ ls
Fabric_project
[ec2-user@ip-172-31-46-52 ~]$
```

Step-5:

result

we can you our beautiful application



Options

Fabric Authentication System

Choose input method:

Capture Webcam

Predict from Webcam

Made with Streamlit

Conclusion

The comparison between machine-made fabric and handmade fabric reveals a nuanced landscape shaped by factors such as technology, craftsmanship, sustainability, and consumer preferences. While machine-made fabrics offer scalability, consistency, and cost-effectiveness, handmade fabrics are valued for their uniqueness, cultural heritage, and artisanal craftsmanship.

There are opportunities to blend modern technology with traditional skills to meet the diverse needs of consumers. Sustainability is also a key focus, driving innovation in both types of fabric production.

Understanding market trends and consumer preferences, the fabric industry can adapt and thrive while preserving the artistry of handmade fabric and leveraging the efficiency of machine-made fabric

References

- https://en.wikipedia.org/wiki/Google_Scholar
- https://www.google.com/search?q=JSTOR%2C&oq=JSTOR%2C&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIJCAEQABgTGIAEMgkIAhAAGBMYgAQyCQgDEAAYExiABDIJCAQQABgTGIAEMgkIBRAAGBMYgAQyCQgGEAAYExiABDIJCAcQABgTGIAEMgkICBAAGBMYgAQyCQgJEAAYExiABNIBCTE0ODJqMGoxNagCALACAA&sourceid=chrome&ie=UTF-8
- https://books.google.co.in/books/about/Textiles.html?id=H7D7ygAACAAJ&redir_esc=y
- https://www.textileebook.com/2020/11/list-of-textile-apparel-and-fashion-ebooks.html
- textileworld.com
- textileexchange.org
- weavespindye.org
- museum.gwu.edu/textile-museum-journal

APPENDIX

VGG-16

```
from tensorflow.keras.applications import VGG16

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import EarlyStopping


# Define paths to your train and validation directories

train_dir = r'C:\NewDataSet\dataset\train'

validation_dir = r'C:\NewDataSet\dataset\valid'


# Define ImageDataGenerator for training and validation data

train_datagen = ImageDataGenerator(

   rescale=1./255,

   rotation_range=20,

   width_shift_range=0.2,

   height_shift_range=0.2,

   shear_range=0.2,

   zoom_range=0.2,

   horizontal_flip=True,

   fill_mode='nearest'

)



validation_datagen = ImageDataGenerator(rescale=1./255)


# Set image dimensions and batch size

img_width, img_height = 224, 224
```

```
batch_size = 32


# Create train and validation generators
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary'
)


validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary'
)
# Load the VGG16 model pre-trained on ImageNet data
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
# Freeze the layers in the base model
for layer in base_model.layers:
    layer.trainable = False
# Add custom classification layers on top of VGG16 with regularization
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.5)(x)  # Add dropout with a rate of 0.5
predictions = Dense(1, activation='sigmoid')(x)  # Binary classification


model = Model(inputs=base_model.input, outputs=predictions)
```

```
# Compile the model
model.compile(optimizer=Adam(lr=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

# Implement Early Stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model with early stopping
epochs = 50  # Adjust the number of epochs as needed
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    callbacks=[early_stopping]
)

# Save the entire model
model.save('vgg16_model.h5')

image_path = "C:/NewDataSet/dataset/test/HandMade/(3).jpg"

# Load the image
img = Image.open(image_path)
img = img.resize((img_width, img_height))  # Resize the image to match model input size

# Convert the image to a numpy array and preprocess it
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.0  # Rescale pixel values

# Get the model's prediction for the image
prediction = model.predict(img_array)
```

```
# Class 1 if prediction >= 0.5, otherwise class 0
predicted_class = 1 if prediction >= 0.5 else 0

# Calculate prediction accuracy
accuracy = prediction[0][0] if predicted_class == 1 else 1 - prediction[0][0]

# Display the image, prediction, and prediction accuracy
plt.imshow(img)
plt.axis('off')
plt.title(f"Prediction: {'Handmade' if predicted_class == 0 else 'Machine-made'}, Accuracy: {accuracy:.2f}")
plt.show()



# Path to the image you want to predict
image_path = "C:/NewDataSet/dataset/test/MachineMade/(23).jpg"

# Load the image
img = Image.open(image_path)
img = img.resize((img_width, img_height))  # Resize the image to match model input size

# Convert the image to a numpy array and preprocess it
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.0  # Rescale pixel values

# Get the model's prediction for the image
prediction = model.predict(img_array)
```

```
# Class 1 if prediction >= 0.5, otherwise class 0

predicted_class = 1 if prediction >= 0.5 else 0


# Calculate prediction accuracy

accuracy = prediction[0][0] if predicted_class == 1 else 1 - prediction[0][0]


# Display the image, prediction, and prediction accuracy

plt.imshow(img)

plt.axis('off')

plt.title(f"Prediction: {'Handmade' if predicted_class == 1 else 'Machine-made'}, Accuracy:
{accuracy:.2f}")

plt.show()
```

Streamlit

```
import streamlit as st

from PIL import Image

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

import cv2

from mpl_toolkits.mplot3d import Axes3D  # Import 3D plotting module


# Load your trained model

model_path = r'C:\Users\anike\Downloads\YO\Dummy\dataset\fabric_authentication_model_New.h5'

model = tf.keras.models.load_model(model_path)


# Function to preprocess the image

def preprocess_image(image):

    img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert from BGR to RGB

    img = cv2.resize(img, (224, 224))  # Resize to model input size

    img = np.expand_dims(img / 255.0, axis=0)  # Normalize and add batch dimension

    return img


# Function to make predictions and display results with a 3D pie chart

def predict_and_display(image):

    predictions = model.predict(image)

    result_prob = predictions[0][0]


    prediction_label = 'Machine-made' if result_prob > 0.5 else 'Hand-made'

    prediction_probability = result_prob if result_prob > 0.5 else 1 - result_prob


    st.write(f"Prediction: {prediction_label} with probability: {prediction_probability:.4f}")


    # Create a 2D pie chart with improved aesthetics
    fig, ax = plt.subplots()
    labels = [f'{prediction_label} ({prediction_probability:.4f})', f'Other ({1 - prediction_probability:.4f})']
    sizes = [prediction_probability, 1 - prediction_probability]
    explode = (0.1, 0)  # explode 1st slice

    colors = ['#ff9999', '#66b3ff']  # Custom colors
    autopct = lambda p: '{:.1f}%'.format(p) if p > 0 else ''  # Hide percentage for zero values

    ax.pie(sizes, explode=explode, labels=labels, autopct=autopct, startangle=90, shadow=True, colors=colors)

    # Equal aspect ratio ensures that pie is drawn as a circle.
    ax.axis('equal')

    # Set a title
    ax.set_title("Fabric Classification")

    st.pyplot(fig)



# Streamlit app
st.title("Fabric Authentication System using Deep Learning")
st.sidebar.title("Options")

# Choose a background color
st.markdown(
    """
    <style>
        .reportview-container {
            background: linear-gradient(to right, #ff6666, #ff8c66, #ffb366, #ffd966, #ffff66, #d9ff66, #b3ff66);
        }
    </style>
    """,
    unsafe_allow_html=True
)

# Option to select between upload and webcam
user_choice = st.selectbox("Choose input method:", ("Upload Image", "Capture Webcam"))

if user_choice == "Upload Image":
    # Upload image functionality
    uploaded_file = st.file_uploader("Choose an image...", type=["jpg", "png", "jpeg"])

    if uploaded_file is not None:
        # Display the uploaded image
        image = Image.open(uploaded_file)
        st.image(image, caption="Uploaded Image", use_column_width=True)
        st.success("Image successfully uploaded!")

        # Button for prediction
        predict_button_upload = st.button("Predict from Uploaded Image")
        if predict_button_upload:
            # Preprocess image and make prediction
            image = np.array(image)
            image = preprocess_image(image)
            predict_and_display(image)

elif user_choice == "Capture Webcam":
    # Display "Predict from Webcam" button only if prediction hasn't been made yet
    predict_button_webcam_clicked = False
    if not predict_button_webcam_clicked:
        predict_button_webcam = st.button("Predict from Webcam")
        if predict_button_webcam:  # Display capture button and prediction upon click
```

```
predict_button_webcam_clicked = True


# Access webcam

cap = cv2.VideoCapture(0)

ret, frame = cap.read()


if ret:

    frame = cv2.flip(frame, 1)  # Flip horizontally for better display

    st.image(frame, channels="BGR", caption="Webcam Image", use_column_width=True)


    # Preprocess image and make prediction

    frame = preprocess_image(frame)

    predict_and_display(frame)


cap.release()

cv2.destroyAllWindows()
```