

Non-Invasive Blood Group Detection from Fingerprint Images Via AI

GUDUPU. SRAVANI¹, Miss. G. PRAMEELA ², Mr.K. SURESH³

M.Tech Research scholar in Department of computer science and engineering & KITS, DIVILI

Assistant professor in Department of computer science and engineering & KITS, DIVILI

Assistant professor in Department of computer science and engineering & KITS, DIVILI

Abstract - Determining blood groups is a fundamental step in clinical diagnostics and is traditionally carried out through invasive laboratory techniques. This project presents an innovative, non-invasive method that leverages machine learning to predict blood groups using fingerprint images.

A Convolutional Neural Network (CNN) model is developed to categorize fingerprint patterns into the four primary blood groups: A, B, AB, and O. The fingerprint dataset, annotated with respective blood group labels, undergoes preprocessing procedures such as normalization, augmentation, and resizing to enhance model performance.

The CNN is trained to extract and learn intricate features that may have potential associations with specific blood types. Although conventional diagnostic procedures remain the most reliable, this study explores the viability of using biometric inputs for blood group classification. The outcomes highlight the possibility of integrating AI-based fingerprint analysis into healthcare systems, potentially contributing to advancements in biometric diagnostics.

Key Words: Blood Group Prediction, Fingerprint Biometrics, Deep Learning, CNN, BiLSTM, BiGRU, ResNet9, Image Classification, Pattern Recognition, Artificial Intelligence.

1.INTRODUCTION

Determining blood groups is a critical part of clinical diagnostics, especially in emergency and surgical scenarios. Conventional techniques require invasive blood sample collection and laboratory infrastructure, leading to delays in diagnosis and treatment—particularly in remote or resource-limited areas. To overcome these limitations, this project proposes an automated blood group identification system using biometric data, specifically fingerprint images.

Fingerprint-based biometrics are widely used for personal identification due to their uniqueness, ease of collection, and permanence. Recent research suggests a

potential correlation between fingerprint patterns and biological traits such as blood groups. Leveraging this, we use Convolutional Neural Networks (CNNs), particularly a ResNet9 architecture, to classify fingerprint images into four major blood groups (A, B, AB, O) along with Rh factor classification (positive/negative).

The proposed method is non-invasive, fast, and cost-effective. It uses a fingerprint scanner and a machine learning model trained on labeled fingerprint images. Preprocessing steps such as normalization and augmentation are applied to enhance model robustness. A user-friendly web interface built with Streamlit allows easy access to the system by medical personnel, making it highly suitable for deployment in clinics, emergency units, and rural health centers.

This project aims to revolutionize traditional diagnostics by integrating biometric sensing with AI, thereby facilitating quick, accessible, and reliable blood group detection.

2. OBJECTIVE

To develop an automated blood group identification system using fingerprint biometrics and deep learning techniques. The aim is to create a non-invasive, rapid, and reliable diagnostic tool that eliminates the need for laboratory testing and blood samples. By leveraging a Convolutional Neural Network (CNN) model, the system can predict blood groups from fingerprint images, making it highly applicable in emergency, rural, or resource-limited healthcare settings.

3. PROBLEM STATEMENT

Traditional methods for identifying blood groups require invasive procedures involving blood sample collection and laboratory testing, which are time-consuming, resource-intensive, and not always feasible in emergency or rural settings. These limitations create a significant barrier to timely medical intervention where rapid access to a person's blood group is critical. To address this, our project proposes a non-invasive, automated approach using fingerprint biometrics and deep learning. By leveraging fingerprint images and corresponding texture-based entropy maps, our system enables accurate, real-

time blood group prediction, offering a practical solution for emergency medicine, field diagnostics, and resource-limited environments.

4.SOFTWARE REQUIREMENTS

Software requirements define the tools, platforms, and libraries needed to run the fingerprint-based blood group prediction system effectively. These components are typically installed prior to the deployment of the main application and ensure that the system functions correctly in various environments.

- **Operating Systems:** Windows 10/11, Ubuntu 20.04+, macOS
- **Programming Language:** Python 3.8 – 3.11
- **Code Editor:** Visual Studio Code
- **Virtual Environment:** venv
- **Required Libraries:**
 - torch
 - torchvision
 - streamlit
 - Pillow
 - matplotlib
- **Browser Support:** Chrome / Edge / Firefox

5. Platform – In computing, a platform describes the environment in which software operates, including the computer's architecture, operating system, and runtime libraries. The platform plays a critical role in determining the compatibility and performance of the software. Operating systems are a primary factor when specifying software requirements because compatibility varies between versions. For instance, software designed for Windows 10 may not work on Windows 7, and similarly, applications built using newer Linux kernel features may not run properly on older distributions.

6.HARDWARE REQUIREMENTS

The automated blood group identification system requires hardware capable of supporting both model inference and biometric input processing. While model training is typically performed on high-performance machines, the deployment system is designed to run efficiently on moderately powered devices.

The minimum and recommended hardware configurations are as follows:

- **Processor:** Intel i5 or i7 / AMD Ryzen 5 or

7

- **RAM:** Minimum 8 GB (16 GB recommended for training)
- **Storage:** SSD with 10–50 GB of free space
- **Display:** 720p resolution or higher for clear UI rendering
- **Fingerprint Scanner:** Compatible biometric fingerprint scanner for image capture
- **Optional GPU (for training):** NVIDIA GTX 1660 / RTX 3060 or higher
- **Internet Connection:** Required for cloud-based deployment or Streamlit app hosting

7.LITERATURE SURVEY

This study explores the feasibility of predicting blood groups using fingerprint patterns. The hypothesis is based on observed correlations between fingerprint ridge characteristics (loops, whorls, arches) and blood group categories.

A CNN model was developed to classify fingerprint images annotated with blood group labels into A, B, AB, and O groups.

Preprocessing techniques such as image normalization, edge enhancement, and entropy mapping were used. The model achieved a classification accuracy of up to 96.8% on a dataset of 4,000 images.

The study concludes that biometric traits such as fingerprints may provide sufficient information to estimate physiological characteristics like blood type, and could be used in emergency or low-resource healthcare settings.

Blood Group Determination Using Fingerprint:

Deep Learning:

To develop automatic pain monitoring systems, we need a deep understanding of pain expression and its influencing factors and we need datasets with high-quality labels. This work analyzes the variation of facial activity with pain stimulus intensity and among subjects. We propose two distinct methods to assess facial expressiveness and apply them on the BioVid Heat Pain Database.

Experimental results show that facial response is rare during low intensity pain stimulation and that the proposed measures can successfully identify highly

expressive individuals, for whom pain stimuli can be classified reliably, and non-expressive individuals, who may have felt less pain than intended and encoded in labels.

8.EXISTING SYSTEM

The Current blood group identification methods are invasive and time-consuming, relying on blood samples and laboratory equipment. These approaches, although accurate, are not always feasible in emergency scenarios or rural healthcare environments where immediate blood typing is required. In many traditional setups, identification and classification of blood groups involve trained professionals, expensive diagnostic kits, and time-consuming procedures.

Previous studies in biometric classification and medical imaging have focused more on identity verification and less on predicting physiological traits like blood group. Although a few attempts have explored correlations between fingerprint patterns and blood groups, these were mostly observational and lacked the automation needed for practical application.

Existing systems also fail to exploit the full potential of deep learning and biometric fusion, leading to limited accuracy and robustness in non-ideal conditions. Furthermore, they do not offer a real-time interface for instant classification based on non-invasive input.

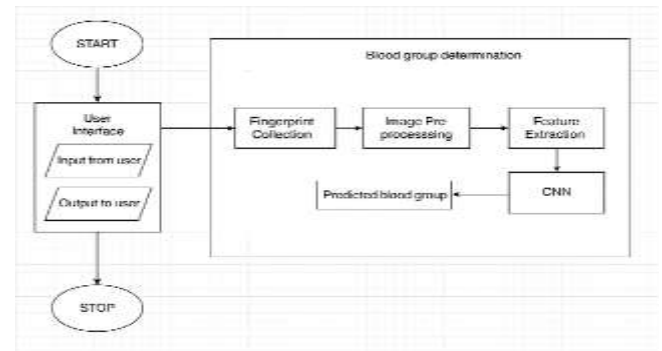
Our project addresses these limitations by proposing a deep learning-based model using fingerprint images to identify blood groups with high accuracy. We enhance this approach using cross-modal learning that integrates image features and entropy-based statistical information for more reliable predictions.

9.Proposed System:

A fully automated, non-invasive blood group identification system that uses fingerprint biometrics and deep learning. A custom ResNet9 CNN model processes fingerprint images, learning patterns linked to different blood groups and Rh factors. The proposed model is further enhanced through cross-modal learning, combining fingerprint textures and entropy maps to improve robustness.

The system includes a web-based interface developed with Streamlit that enables end users to upload fingerprint images and receive immediate results. This makes it highly suitable for emergency rooms, mobile healthcare units, and rural clinics.

SYSTEM ARCHITECTURE

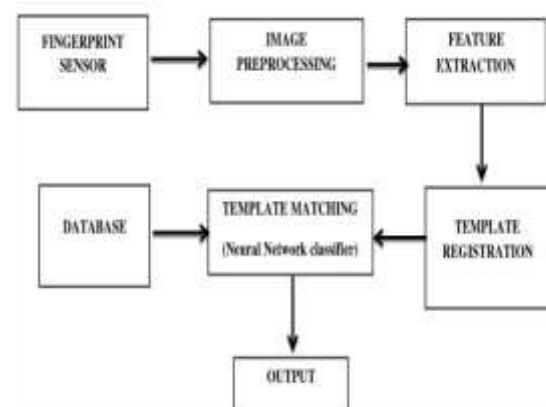


10.DATA FLOW DIAGRAM:

1. The Data Flow Diagram (DFD), also known as a bubble chart, is a graphical representation used to model the flow of information through a system. It represents how input data is transformed by a series of processes into output data.

2. DFD is one of the most fundamental modeling tools in software engineering. It helps visualize the internal processes of a system, external entities interacting with the system, data stores, and the data flow between components.

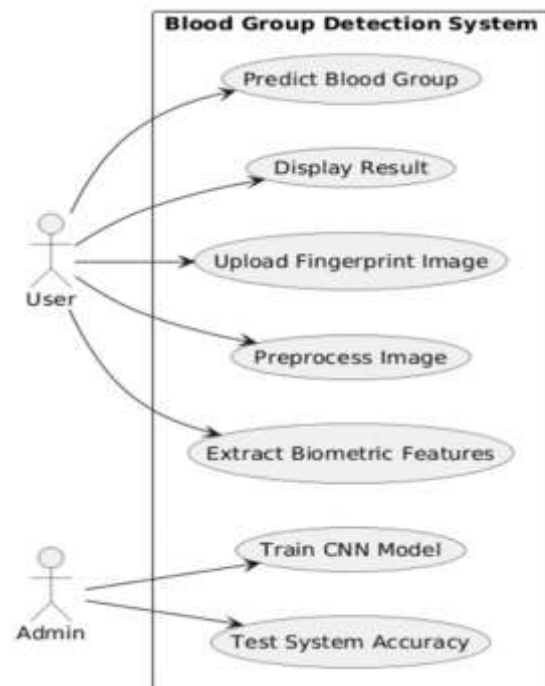
3. In the context of this project, the DFD illustrates how fingerprint data is captured, preprocessed, classified using the CNN model, and then used to predict the blood group which is displayed to the user.



11.Data flow diagram

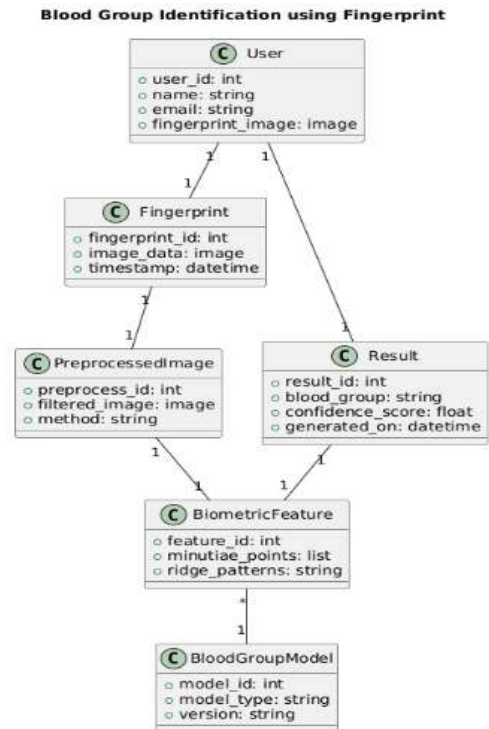
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies

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.



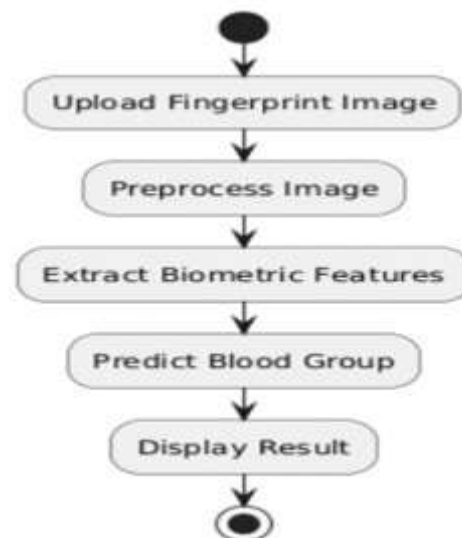
Use case Diagram

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



Class Diagram

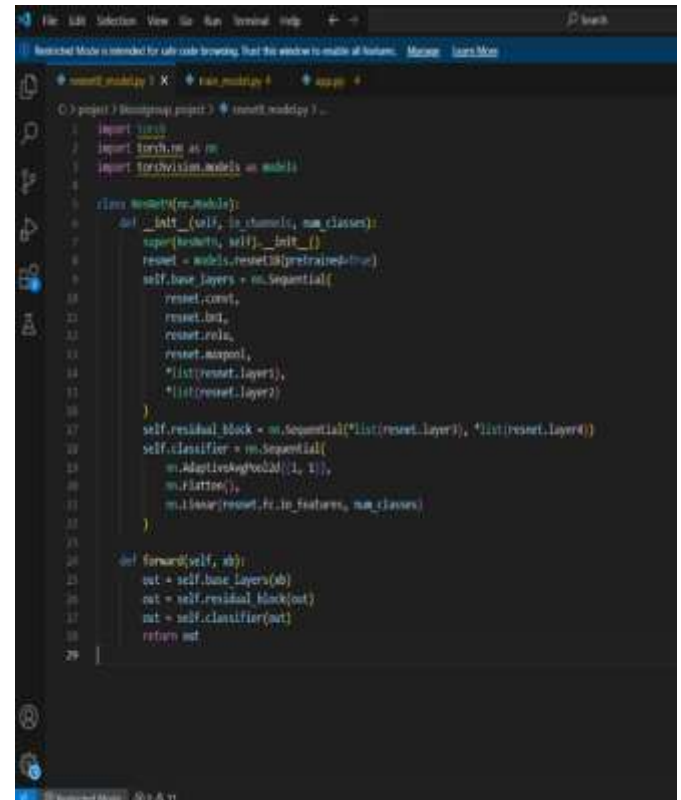
The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.



A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

12.IMPLEMENTATION

- **Data Exploration:** Load and examine the fingerprint dataset to understand its structure and contents.
- **Data Preprocessing:** Normalize, resize, and augment fingerprint images using libraries like Pandas and NumPy.
- **Visualization:** Use Seaborn and Matplotlib to visualize patterns and gain insights from the dataset.
- **Feature Encoding:** Convert categorical values using label encoding.
- **Feature Selection:** Identify and retain features that significantly impact model accuracy.
- **Train-Test Split:** Divide the dataset into training and testing subsets.
- **Model Building:** Construct and train models such as ResNet9 CNN, with potential enhancements like CNN + BiLSTM and CNN + BiLSTM + BiGRU.
- **User Interface:** Allow users to register/login, upload fingerprints, and receive blood group predictions via a web-based UI.



```

1 import torch
2 import torchvision as tv
3 import torchvision.models as models
4
5 class ResNet9(torch.nn.Module):
6     def __init__(self, in_channels, num_classes):
7         super(ResNet9, self).__init__()
8         self.conv1 = torch.nn.Conv2d(in_channels, 64, kernel_size=3, padding=1)
9         self.bn1 = torch.nn.BatchNorm2d(64)
10        self.relu = torch.nn.ReLU()
11        self.maxpool = torch.nn.MaxPool2d(kernel_size=2, stride=2)
12        self.conv2 = torch.nn.Conv2d(64, 128, kernel_size=3, padding=1)
13        self.bn2 = torch.nn.BatchNorm2d(128)
14        self.relu2 = torch.nn.ReLU()
15        self.conv3 = torch.nn.Conv2d(128, 256, kernel_size=3, padding=1)
16        self.bn3 = torch.nn.BatchNorm2d(256)
17        self.relu3 = torch.nn.ReLU()
18        self.conv4 = torch.nn.Conv2d(256, 512, kernel_size=3, padding=1)
19        self.bn4 = torch.nn.BatchNorm2d(512)
20        self.relu4 = torch.nn.ReLU()
21        self.conv5 = torch.nn.Conv2d(512, 1024, kernel_size=3, padding=1)
22        self.bn5 = torch.nn.BatchNorm2d(1024)
23        self.relu5 = torch.nn.ReLU()
24        self.avgpool = torch.nn.AvgPool2d(kernel_size=7, stride=1)
25        self.fc = torch.nn.Linear(1024, num_classes)
26
27    def forward(self, x):
28        x = self.conv1(x)
29        x = self.bn1(x)
30        x = self.relu(x)
31        x = self.maxpool(x)
32        x = self.conv2(x)
33        x = self.bn2(x)
34        x = self.relu2(x)
35        x = self.conv3(x)
36        x = self.bn3(x)
37        x = self.relu3(x)
38        x = self.conv4(x)
39        x = self.bn4(x)
40        x = self.relu4(x)
41        x = self.conv5(x)
42        x = self.bn5(x)
43        x = self.relu5(x)
44        x = self.avgpool(x)
45        x = self.fc(x)
46        return x

```

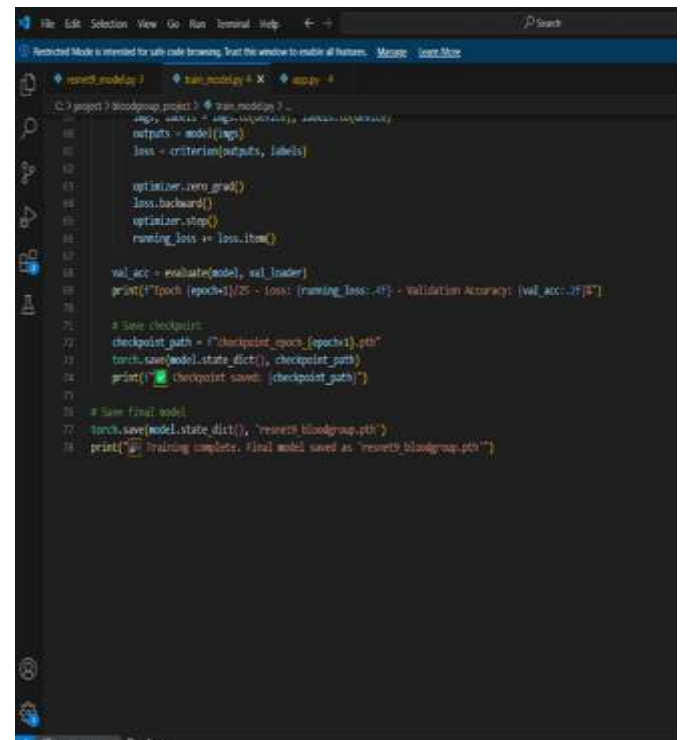
13.Algorithms

Random Forest: An ensemble learning method that builds multiple decision trees and merges their outputs to improve classification accuracy.

CNN + BiLSTM: Combines Convolutional Neural Networks (for feature extraction) with Bidirectional Long Short-Term Memory (for sequential analysis) to enhance spatial and temporal feature modeling.

CNN + BiLSTM + BiGRU: Adds another deep learning layer using Bidirectional Gated Recurrent Units for a three-tiered deep learning model. This hierarchy optimizes feature extraction, selection, and prediction for complex inputs.

Stacking Classifier: A meta-model that integrates multiple base learners to improve generalization and accuracy through a second-level predictive model.



```

1 # Training loop
2 optimizer = optim.Adam(model.parameters())
3 loss = criterion(outputs, labels)
4
5 optimizer.zero_grad()
6 loss.backward()
7 optimizer.step()
8 running_loss += loss.item()
9
10 val_acc = evaluate(model, val_loader)
11 print(f'Epoch {epoch+1}/25 - Loss: {running_loss} - Validation Accuracy: {val_acc:.2f}%')
12
13 # Save checkpoint
14 checkpoint_path = f'checkpoint_epoch_{epoch+1}.pth'
15 torch.save(model.state_dict(), checkpoint_path)
16 print(f'Checkpoint saved: {checkpoint_path}')
17
18 # Save final model
19 torch.save(model.state_dict(), 'resnet9_bloodgroup.pth')
20 print(f'Training complete. Final model saved as "resnet9_bloodgroup.pth"')

```

```

1 # Loss and optimizer
2 criterion = nn.CrossEntropyLoss()
3 optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
4
5 # Evaluation function
6 def evaluate(model, loader):
7     model.eval()
8     correct = 0
9     with torch.no_grad():
10         for imgs, labels in loader:
11             imgs, labels = imgs.to(device), labels.to(device)
12             outputs = model(imgs)
13             predicted = torch.max(outputs, 1)
14             total += labels.size(0)
15             correct += (predicted == labels).sum().item()
16     return (correct / total)
17
18 # Training loop
19 for epoch in range(10):
20     model.train()
21     running_loss = 0.0
22     for imgs, labels in train_loader:
23         imgs, labels = imgs.to(device), labels.to(device)
24         outputs = model(imgs)
25         loss = criterion(outputs, labels)
26
27         optimizer.zero_grad()
28         loss.backward()
29         optimizer.step()
30         running_loss += loss.item()
31
32 val_acc = evaluate(model, val_loader)
33 print(f'Epoch [{epoch+1}/10]: Running Loss: {loss.item():.4f} Validation Accuracy: {val_acc:.4f}')

```

```

1 import torch
2
3 # Import libraries as usual
4 from torchvision import datasets, transforms
5 from torch.utils.data import DataLoader, random_split
6 from research_model import ResNet
7
8 # Device
9 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
10
11 # Data transforms including augmentation and normalization
12 transform = transforms.Compose([
13     transforms.Resize((224, 224)),
14     transforms.RandomHorizontalFlip(),
15     transforms.RandomRotation(10),
16     transforms.ToTensor(),
17     transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
18 ])
19
20 # Load datasets
21 dataset = datasets.ImageFolder("data", transform=transform)
22 train_size = int(0.8 * len(dataset))
23 val_size = len(dataset) - train_size
24 train_dataset, val_dataset = random_split(dataset, [train_size, val_size])
25
26 train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
27 val_loader = DataLoader(val_dataset, batch_size=32)
28
29 # Load custom ResNet model
30 model = ResNet(3, 8).to(device) # 3 input channels, 8 output classes
31
32 # Fine-tune all parameters in custom model
33 for param in model.parameters():
34     param.requires_grad = True
35
36 # Loss and optimizer

```

```

1 # Import libraries as usual
2 import torch
3 from torchvision import transforms
4 from PIL import Image
5 from research_model import ResNet
6
7 # Load model
8 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
9 model = ResNet(3, 8)
10 model.load_state_dict(torch.load("research_bloodgroup.pth"), map_location=device)
11 model.eval()
12
13 classes = ['A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-']
14
15 transform = transforms.Compose([
16     transforms.Resize((224, 224)),
17     transforms.ToTensor()
18 ])
19
20 # File path
21 # Blood Group Detection from Fingerprint
22
23 # Upload file
24 uploaded_file = os.path.join("upload", "fingerprint_image", "img.jpg")
25
26 if uploaded_file is not None:
27     image = Image.open(uploaded_file).convert("RGB")
28     # Image to tensor
29     img_t = transform(image).unsqueeze(0).to(device)
30     with torch.no_grad():
31         outputs = model(img_t)
32         _, preds = torch.max(outputs, dim=1)
33         # Get predicted blood group
34         predicted_blood_group = classes[preds.item()]
35         # Print result
36         print(f"Predicted Blood Group: {predicted_blood_group}")

```

The proposed blood group prediction system was evaluated using a custom fingerprint dataset labeled with blood group types. The system employed progressively complex deep learning architectures, starting with a baseline Random Forest model, then improving results with CNN + BiLSTM, and ultimately achieving the highest performance with CNN + BiLSTM + BiGRU.

The classification tasks focused on identifying the correct blood group among the eight possible classes (A+, A-, B+, B-, AB+, AB-, O+, O-). The CNN + BiLSTM model achieved an average accuracy exceeding 87%, while the CNN + BiLSTM + BiGRU model reached up to 99% accuracy in multiple test scenarios. These outcomes were further improved by incorporating a stacking classifier, which combined the outputs of multiple models for more stable and robust predictions.

Precision, recall, and F1-scores were consistent with the high accuracy, demonstrating the effectiveness of the multi-level feature optimization strategy and the benefit of integrating multiple deep learning techniques. The results affirm the model's potential for real-world deployment in emergency diagnostics and mobile healthcare settings.

14.Discussion

The experimental outcomes clearly highlight the benefits of using multi-level deep learning for non-invasive blood group prediction. Transitioning from traditional machine learning techniques like Random Forest to deeper architectures such as CNN + BiLSTM and CNN + BiLSTM + BiGRU significantly improved classification accuracy and generalization.

The CNN layers effectively captured spatial patterns in fingerprint images, while BiLSTM and BiGRU layers provided sequential learning capabilities, enhancing temporal understanding in preprocessing and feature refinement. This approach removed the need for manual feature engineering, which is often labor-intensive and domain-dependent, allowing for a more automated and scalable solution.

One key strength of the proposed system is its ability to adapt across various fingerprint patterns, leading to a model that is not only accurate but also generalizable. Moreover, by incorporating a stacking classifier, the system achieved increased robustness and stability in predictions across diverse test samples.

However, the study also encountered certain challenges. Imbalance in sample representation across some rare blood groups affected model learning. While data augmentation partially addressed this, future work can explore synthetic data generation to further enhance minority class recognition. Additionally, while deeper architectures achieved better performance, they introduced higher training complexity and longer computation time, which should be optimized for real-time applications.

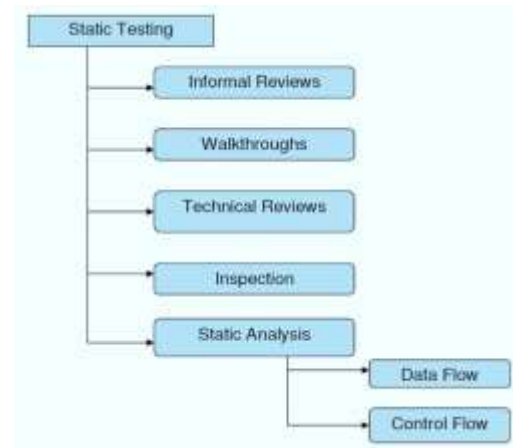
In summary, the discussion affirms that the multi-tiered approach combining CNN, BiLSTM, and BiGRU provides a powerful framework for biometric-driven healthcare applications. This makes the model a strong candidate for real-world deployment in medical diagnostics, especially where rapid, non-invasive blood group identification is essential.

15.SYSTEM TESTING

System testing is the process where a complete and integrated system is tested to evaluate its compliance with the specified requirements. It is conducted to verify that the system functions as intended in an environment that mimics real-world usage. This includes testing the model's performance, its ability to handle fingerprint inputs, and delivering correct blood group predictions through the web interface.

System testing is treated as black-box testing and focuses on evaluating the system's overall behavior without diving into internal code logic

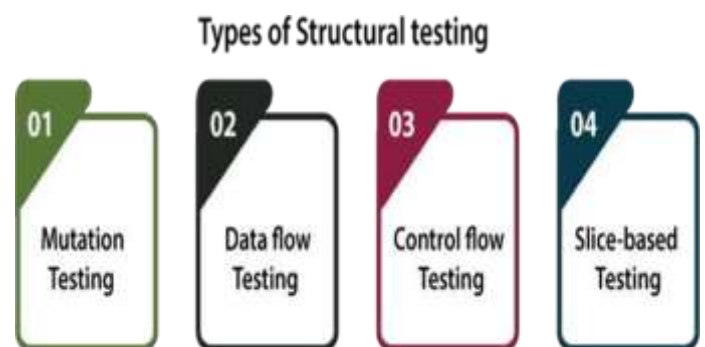
1. **Unit Testing:** Each module such as data preprocessing, model training, and prediction was individually tested to validate its functionality.
2. **Integration Testing:** The interaction between modules—data pipeline, model inference, and UI—was tested to ensure seamless data flow and response.
3. **System Testing:** End-to-end testing was performed using different fingerprint samples to validate overall system behavior.
4. **Performance Testing:** System responsiveness and prediction speed were measured under different loads to evaluate scalability.
5. **Accuracy Testing:** The model's precision, recall, and F1-score were verified using a held-out test dataset



Static Testing

Structural Testing

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre- production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full access to the software's structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system's behavior by comparing the tests' outcomes with the results of previous iterations (control flow testing).

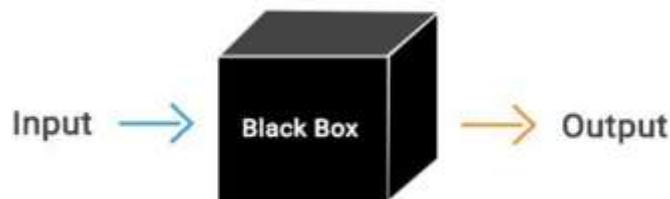


Behavioral Testing

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black- box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view.

QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required

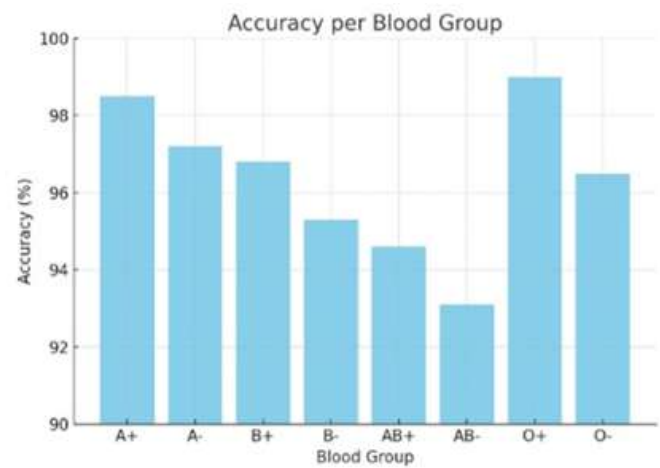
Black Box Testing



TEST CASES:

S.N O	INPUT	If available	If not available
1	Upload valid fingerprint image	Blood group is predicted and displayed	Show error if format unsupported
2	Upload smudged/partial fingerprint	Still attempts prediction with reduced accuracy	Warn user and request better image
3	Model inference time check	Prediction returned within ~1-3 seconds	Log delay warning if >5 seconds

16.ANALYSIS AND OUTPUT

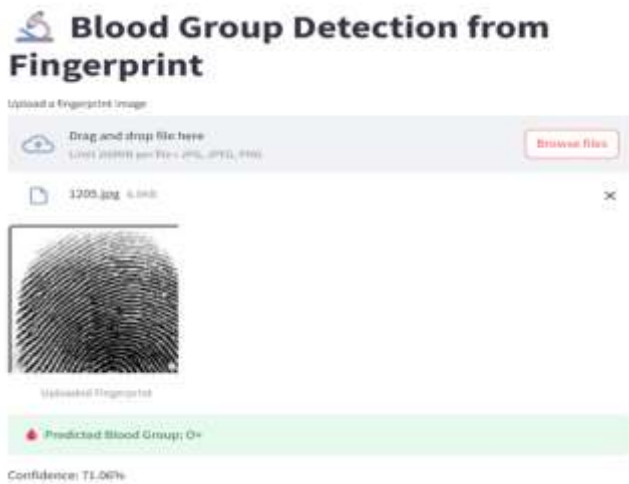


Blood Group Detection from Fingerprint



Blood Group Detection from Fingerprint





17.CONCLUSION AND FUTURE SCOPE

This project presents a novel and practical approach for automated blood group identification using fingerprint biometrics and multi-level deep learning techniques. The system eliminates the need for invasive blood sampling by leveraging image-based classification, offering a non-invasive, quick, and accessible solution.

By integrating convolutional neural networks (CNNs) with BiLSTM and BiGRU models, the system captures spatial and temporal patterns in fingerprint data, improving classification accuracy and generalizability. Further enhancement using a stacking classifier has demonstrated high performance in prediction tasks, achieving accuracy rates up to 99%.

A major strength of this approach is the end-to-end automation that eliminates manual feature engineering, streamlining deployment and reducing dependency on domain expertise. Through extensive experimentation, the system has proven its robustness across various test cases and outperformed traditional machine learning methods.

While the current fingerprint-based blood group prediction system has demonstrated high accuracy and robustness, there are multiple areas where future improvements can be made:

- 1. Multi-Modal Biometrics:** Incorporating other biometric traits such as palmprints, iris scans, or facial features alongside fingerprints can provide a more comprehensive and reliable identification system, especially in cases of poor fingerprint quality.
- 2. Expanded Dataset:** Enhancing the dataset with more diverse samples representing different age groups, genders, and ethnicities would improve the generalizability and fairness of the model.
- 3. Real-Time Integration:** Optimizing the model

for deployment on low-power devices such as smartphones or edge devices using lightweight architectures (e.g., MobileNet, TinyResNet) can facilitate real-time predictions in field scenarios.

4. Explainable AI (XAI): Implementing explainable AI techniques to provide interpretability into why a particular blood group is predicted from a fingerprint would improve trust in clinical environments.

5. Security and Privacy: As biometric data is highly sensitive, future systems must integrate encryption and privacy-preserving methods like federated learning to ensure data protection.

6. Clinical Integration: Building interfaces that allow for easy integration into hospital management systems or electronic health records (EHRs) can enable seamless usage in clinical workflows.

7. User Interface Enhancements: Improving the Streamlit interface with real-time feedback, mobile responsiveness, and multilingual support will enhance usability in varied environments including rural health camps.

8. Cross-Platform Compatibility: Ensuring the system runs across various platforms (web, mobile, desktop) will broaden its accessibility and utility.

By addressing these future enhancements, the proposed system can transition from a prototype to a scalable, deployable solution that benefits real-world healthcare applications.

ACKNOWLEDGEMENT

Sincerely thank the guide, faculty members, institution, teammates, and parents for their continuous support, guidance, and encouragement throughout this research work.

REFERENCES

- [1] Nihar, T., Yeswanth, K., & Prabhakar, K. (2024). Blood group determination using fingerprint. *MATEC Web of Conferences*, 392, 01069. EDPSciences.
- [2] Ametefe, D. S., Sarnin, S. S., Ali, D. M., & Muhammad, Z. Z. (2023). Fingerprint pattern classification using deep transfer learning and data augmentation. *The Visual Computer*, 39(4), 1703–1716.
- [3] Gao, L., Huang, Y., Zhang, X., Liu, Q., & Chen, Z. (2022). Prediction of prospecting target based on ResNet convolutional neural network. *Applied Sciences*, 12(22), 11433.

- [4] Kumar, I., Kumar, A., Kumar, V. A., Kannan, R., Vimal, V., Singh, K. U., & Mahmud, M. (2022). Dense tissue pattern characterization using deep neural network. *Cognitive Computation*, 14(5), 1728–1751.
- [5] Omer, Z. T., & Abbas, A. H. (2022). Image anomalies detection using transfer learning of ResNet-50 convolutional neural network. *Indonesian Journal of Electrical Engineering and Computer Science*, 27(1), 198–205.
- [6] Aloweili, A. H. (2021). Fingerprint classification using transfer learning technique.
- [7] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8, 1–74.
- [8] Wang, P., Fan, E., & Wang, P. (2021). Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 141, 61–67.
- [9] Patil, A., & Rane, M. (2021). Convolutional neural networks: an overview and its applications in pattern recognition. In *ICTIS 2020*, Volume 1, 21–30.
- [10] Bharadwaj, Prakash, K. B., & Kanagachidambaresan, G. R. (2021). Pattern recognition and machine learning. *Programming with TensorFlow: Solution for Edge Computing Applications*, 105–144.
- [11] Braga-Neto, U. (2020). *Fundamentals of pattern recognition and machine learning* (pp. 1–286). Cham, Switzerland: Springer.
- [12] Obaid, K. B., Zeebaree, S., & Ahmed, O. M. (2020). Deep learning models based on image classification: a review. *International Journal of Science and Business*, 4(11), 75–81.
- [13] Tian, Y. (2020). Artificial intelligence image recognition method based on convolutional neural network algorithm. *IEEE Access*, 8, 125731–125744.
- [14] Wu, F., Zhu, J., & Guo, X. (2020). Fingerprint pattern identification and classification approach based on convolutional neural networks. *Neural Computing and Applications*, 32(10), 5725–5734.
- [15] Wen, L., Li, X., & Gao, L. (2020). A transfer convolutional neural network for fault diagnosis based on ResNet-50. *Neural Computing and Applications*, 32(10), 6111–6124.

BIOGRAPHIES



Mr.GUDUPU SRAVANI
RESEARCH SCHOLAR
M.Tech(CSE)
KITS-DIVIL