**Project Report**

# Blood Cancer Detection System

*Project Title:  Blood Cancer Detection*

*Subtitle: Machine Learning / Deep Learning*

*Name: Utkarsh Kushwaha[25BEC10051]*

*Institution : VIT Bhopal*

*Course:Fundamentals Of AI and ML*

## 2. Introduction

Blood cancer, particularly leukemia, involves the abnormal proliferation of white blood cells and is a serious hematological disease. Traditional diagnostic methods rely on manual microscopic examination of blood smears by expert hematologists, which can be slow, subjective, and prone to human error. With advancements in machine learning and deep learning, we can automate the detection of cancerous cells in blood smear images, improving speed, consistency, and potentially early diagnosis. This project aims to build a system that takes stained blood smear images, processes them, and classifies whether they indicate leukemia or not using a trained deep learning model.

---

## 3. Problem Statement

- **Primary Problem:** How to automatically and accurately detect leukemia (blood cancer) from microscopic blood smear images?
- **Challenges:**
    - High variability in cell morphology, staining, and imaging conditions.
    - Class imbalance: cancerous cell images may be fewer than healthy ones.
    - Need for high sensitivity (i.e., low false negatives) to avoid missing cancer.
    - Requirement of model interpretability and clinical trust.
    - Deployment in resource-constrained settings (labs with limited compute).

---

## 4. Functional Requirements

1. **Image Input:** Accept blood smear images (e.g., JPEG, PNG) uploaded by the user.
2. **Preprocessing:** Normalize, resize, augment, and optionally segment cell regions.
3. **Feature Extraction / Model Inference:** Use a trained CNN (or hybrid model) to extract features and infer classification.
4. **Classification:** Predict class labels (e.g., “Leukemia / Normal” or more detailed subtypes).
5. **Confidence Score:** Provide probability or confidence for predictions.
6. **Results Display:** Show input image, classification result, and confidence.
7. **Report Generation:** Generate a downloadable report with prediction, metrics, and possibly marked image regions.
8. **Model Training / Retraining:** Optionally allow adding new labeled images and retraining / fine-tuning.
9. **Performance Metrics:** Compute and display metrics such as accuracy, precision, recall, F1-score, confusion matrix.
10. **User Management (if multi-user):** Allow different roles (e.g., lab technician, clinician) to upload, view, and manage predictions.

11. **Logging & Storage:** Store images, predictions, and logs for audit and future training.

---

## 5. Non-functional Requirements

- **Accuracy:** The model should maintain high predictive accuracy (target: > 90%).
- **Robustness:** Should handle variation in image quality, staining, and noise.
- **Speed / Latency:** Predictions should be generated in reasonable time (e.g., < 5 seconds per image on deployment hardware).
- **Scalability:** The system should scale to handle large numbers of images and users.
- **Usability:** The UI should be simple and intuitive for non-technical users (lab technicians, clinicians).
- **Security & Privacy:** Sensitive patient image data should be securely stored and protected (e.g., with encryption, access controls).
- **Portability:** The system should be deployable on local machines (e.g., in a lab) or on a cloud server.
- **Maintainability:** Codebase should be modular, allowing future updates or swapping of the model.
- **Explainability:** The model should provide interpretability (e.g., highlight areas of the image that influenced the decision).
- **Compliance:** The system should meet relevant medical data regulations / standards (depending on region).

---

## 6. System Architecture

Here is a high-level architecture for the system:

- **Data Layer:**
  - Raw blood smear images (storage)
  - Labeled dataset (normal vs leukemia)
  - Database for storing images, predictions, user data (optional)
- **Preprocessing Module:**
  - Image normalization (intensity, color)
  - Resizing, scaling
  - Data augmentation (rotate, flip, zoom)
  - (Optional) Segmentation of cells or nuclei
- **Model / Inference Module:**
  - CNN-based classifier (or other deep learning architecture)
  - Feature extractor (if hybrid)
  - Prediction module
- **Evaluation Module:**
  - Metrics calculator (accuracy, precision, recall, F1)
  - Confusion matrix generator

    - (Optional) Explainability: Grad-CAM or saliency mapping
- **User Interface:**
    - Web frontend / Desktop UI
    - Image upload page
    - Prediction results page
    - Report download page
- **Backend / Server:**
    - API endpoints for upload, inference, model retraining
    - Logging and storage service
    - Authentication & authorization (if multi-user)
- **Storage / Database:**
    - Stores user data, images, and prediction results
    - Model storage (versioning)
- **Deployment:**
    - Cloud service (e.g., AWS, GCP) or local server
    - GPU (if needed) or CPU inference environment

---

## 7. Design Diagrams

### Use Case Diagram

- **Actors:** Lab Technician, Clinician / Pathologist, System
- **Use Cases:**
    - Upload Blood Smear Image
    - Preprocess Image
    - Classify Image
    - View Prediction & Confidence
    - Generate Report
    - Retrain / Fine-tune Model
    - View Performance Metrics

### Workflow Diagram

1. Technician collects / scans a blood smear slide.
2. Technician uploads the image via UI.
3. System preprocesses the image (normalization, augmentation, segmentation).
4. Preprocessed image is passed to the model for inference.
5. Model returns a prediction (normal / leukemia + confidence).
6. UI displays results, including confidence and optionally heatmap / explainability.
7. Prediction and associated data are saved to storage.
8. (Optional) Clinician reviews and confirms.
9. (Optional) Retraining: New labeled images are fed back for model update.

### Sequence Diagram

```
Lab Technician → UI: Upload(image)
UI → Preprocessor: send(image)
Preprocessor → Model: processed_image
Model → Predictor: prediction + confidence
Predictor → UI: display result
UI → Storage: save(image, prediction, confidence)
```

**Class / Component Diagram**

- `ImageUploader` (handles image uploads)
- `ImagePreprocessor` (normalizes, resizes, augments)
- `Segmenter` (optional; isolates WBCs / nuclei)
- `CNNClassifier` (deep model)
- `Predictor` (runs inference + returns class + confidence)
- `ExplainabilityModule` (Grad-CAM / saliency)
- `ResultReporter` (report generation)
- `UserInterface` (frontend)
- `StorageManager` (handles database or file storage)
- `ModelTrainer` (for retraining / fine-tuning)
- `MetricEvaluator` (computes metrics)

**ER Diagram (if using storage)**

- **User**: `user_id`, `name`, `role` (technician / clinician)
- **Image**: `image_id`, `user_id (FK)`, `file_path`, `upload_time`
- **Prediction**: `prediction_id`, `image_id (FK)`, `predicted_label`, `confidence`, `timestamp`
- **ModelVersion**: `version_id`, `trained_on_dataset_info`, `accuracy`, `precision`, `recall`, `F1`

---

## 8. Design Decisions & Rationale

- **Why deep learning (CNN)?**
  CNNs are very effective for image data because they can automatically learn hierarchical features (edges → shapes → textures) that are useful for distinguishing normal vs cancerous cells. Many existing studies use CNNs for leukemia detection. IJRASET+2MDPI+2
- **Preprocessing & Data Augmentation:**
  To increase robustness to variations in staining, lighting, and image capture, augmentation (rotations, flips, zooms) is used.
- **Segmentation:**
  Optionally segmenting white blood cells or nuclei helps focus the classifier on relevant regions, reducing noise from background.
- **Feature Selection / Hybrid Model (if used):**
  If combining deep features with ML classifiers (like SVM), one can extract deep features

and then select the most discriminative ones. For example, ResRandSVM approach uses feature selection + SVM for leukemia classification. PubMed+1

- **Explainability:**
  Using tools like Grad-CAM offers interpretability, which is important in medical applications for clinician trust.
- **Storage & Retraining:**
  Maintaining a database of images, predictions, and model versions supports retraining and audit—critical for a clinical environment.
- **UI / Deployment Choice:**
  A web interface ensures ease of use and cross-platform accessibility; deploying on a server (cloud or local) allows scaling and central management.

---

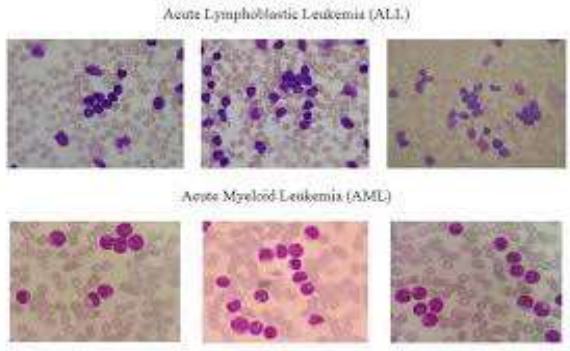
## 9. Implementation Details

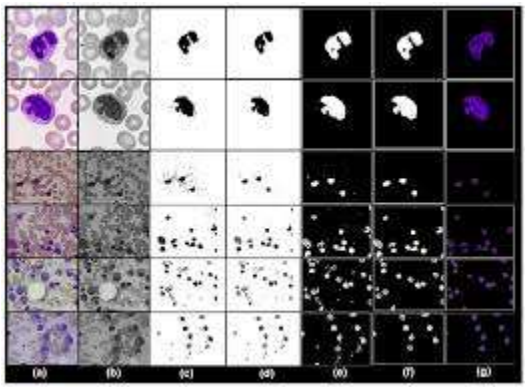
- **Language & Frameworks:** Python, using TensorFlow / Keras or PyTorch.
- **Dataset:** (Specify the dataset you are using — e.g., public blood smear image dataset)
- **Preprocessing Steps:**
  1. Load image
  2. Resize to fixed input size (e.g., 224×224)
  3. Normalize pixel values (e.g., scale to [0, 1])
  4. Data augmentation: rotate up to ±20°, flip horizontally/vertically, zoom ±10%, shift.
  5. (Optional) Segment cells using morphological operations or clustering.
- **Model Architecture:**
  - Option A: Custom CNN (conv → pooling → conv → pooling → dense → softmax)
  - Option B: Pre-trained CNN (e.g., ResNet50, MobileNet) + fine-tuning
- **Training:**
  - Split dataset: training, validation, test
  - Loss: binary cross-entropy (if binary classification) or categorical cross-entropy
  - Optimizer: Adam / SGD
  - Hyperparameters: batch size = 32, epochs = 50 (can vary)
- **Evaluation:**
  - Metrics: Accuracy, Precision, Recall, F1-score, Confusion Matrix
  - Use validation set for hyperparameter tuning
  - Save best-performing model version
- **Explainability:**
  - Use Grad-CAM or saliency map to highlight important image regions for each prediction
- **Deployment / UI:**
  - Build a simple web app (Flask or Django)
  - Endpoints: `/upload`, `/predict`, `/report`
  - Frontend: HTML + Bootstrap for image upload and result display
- **Storage / Logging:**

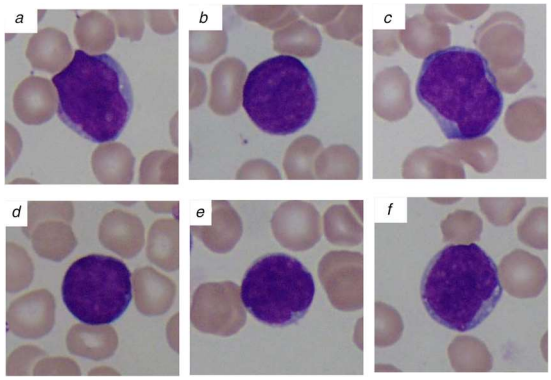
    - Use SQLite / PostgreSQL or file storage for images
    - Log predictions: `image_id`, `prediction`, `confidence`, `timestamp`
- **Model Retraining:**
    - A retraining pipeline that loads new labeled images, re-trains (or fine-tunes) the model, and versions it.

---

## 10. Screenshots / Results






---

## 11. Testing Approach

1. **Train-Test Split Testing:** Use independent test set not seen during training.
2. **Cross-Validation:** Use k-fold cross-validation (e.g., 5-fold) to check model stability.
3. **Robustness Testing:** Test on images with varying quality (stain variation, blur, noise) to check system's robustness.
4. **Performance Metrics:** Evaluate using accuracy, precision, recall, F1, confusion matrix.
5. **User Acceptance Testing (UAT):** Involve a lab technician / pathologist to test usability, correctness of result, and interpretability.
6. **Error Analysis:** Identify false positives and false negatives, inspect those images, and check reasons for misclassification.
7. **Retraining Validation:** After retraining with new data, check if performance improves or remains consistent.

---

## 12. Challenges Faced

- **Data Variability:** Differences in staining, illumination, and focus made preprocessing challenging.
- **Class Imbalance:** If leukemia images are fewer, the model may overfit to the majority class.
- **Overfitting:** With a limited dataset, the model risks over-fitting; needed strong regularization or augmentation.
- **Segmentation Difficulty:** Segmenting WBCs / nuclei reliably is non-trivial; noise and overlapping cells complicate things.
- **Computation Constraints:** Training deep CNNs requires GPUs; limited hardware increased training time.
- **Explainability:** Generating meaningful explainability maps (e.g., Grad-CAM) and interpreting them for clinicians was challenging.
- **Deployment Complexity:** Integrating model, UI, storage, and retraining pipeline reliably posed engineering challenges.
- **Data Privacy:** Ensuring patient image data is stored securely and used ethically.

---

## 13. Learnings & Key Takeaways

- Deep learning (especially CNNs) is very effective for medical image classification tasks like leukemia detection.
- Preprocessing and data augmentation play a vital role in improving model generalization.
- Working with domain experts (hematologists) helps validate model predictions and interpretability.
- Explainability (e.g., Grad-CAM) is important for trust in medical AI systems.
- Building a full system (not just the model) — including UI, storage, retraining, and evaluation — is essential to make it usable in real-world settings.

- Regular evaluation, error analysis, and retraining are critical to maintain and improve performance.

---

## 14. Future Enhancements

- **Subtype Classification:** Extend the classifier to detect leukemia subtypes (e.g., ALL, AML, CLL) rather than only binary detection.
- **Segmentation Network:** Use an advanced segmentation architecture like U-Net to more precisely isolate WBCs or nuclei.
- **Ensemble Models:** Use ensemble methods (multiple CNNs / hybrid) to improve robustness.
- **Transfer Learning:** Use pre-trained networks (e.g., EfficientNet, ResNet) for better feature extraction.
- **Explainability & Interpretability Enhancements:** Add more interpretability methods (e.g., LIME, SHAP) for model decisions.
- **Mobile / Edge Deployment:** Deploy the system on mobile devices or edge hardware for use in low-resource labs.
- **Clinical Validation:** Run a clinical trial / pilot study with real-world lab data to validate the system.
- **Automated Feedback Loop:** Implement an automated feedback loop where pathologist corrections / confirmations can be used to fine-tune the model.
- **Data Expansion:** Collect and integrate datasets from multiple labs / staining protocols to improve generalizability.

---

## 15. References


- Zare, L., Rahmani, M., Khaleghi, N., Sheykhivand, S., & Danishvar, S. (2024). Automatic Detection of Acute Leukemia (ALL and AML) Utilizing Customized Deep Graph Convolutional Neural Networks. *Bioengineering*, **11(7)**. MDPI
- Keerthivasan, S. P., & Saranya, N. (2023). Acute Leukemia Detection using Deep Learning Techniques. *International Research Journal on Advanced Science Hub*. rspsciencehub.com
- ResRandSVM: Hybrid Approach for Acute Lymphocytic Leukemia Classification in Blood Smear Images. *Diagnostics*. MDPI+1
- Accurate Machine-Learning-Based Classification of Leukemia from Blood Smear Images. *PubMed*. PubMed
- Detection of Acute Lymphoblastic Leukemia using Image Segmentation and Data Mining Algorithms. *PubMed*. PubMed
- Zolfaghari, M., & Sajedi, H. (2023). A survey on automated detection and classification of acute leukemia and WBCs in microscopic blood cells. *arXiv*. arXiv