Type you

Project Phase Template

Project Title:

Hematovision: Advanced Blood Cells Classification Using Transfer Learning

Team Members:

TEAM LEADER: MEkALA SIVA NAGA

TEAM MEMBER : KAMINENI CHAMUNDESWARI

TEAM MEMBER : MADDELA KIRAN SAI BABU

TEAM MEMBER : ANNAMDEVARA SAISRI

Phase-1: Brainstorming & Ideation

•Objective:

Clearly define the problem space within the context of healthcare diagnostics. Understand the inefficiencies of current manual processes in hematology. Align project goals with real-world needs by interviewing domain experts or studying literature. Explore the role of machine learning in medical imaging, specifically focusing on transfer learning's advantages in small datasets.

•Key Points:

1. Problem Statement:

•Accurate blood cell classification is essential for early disease diagnosis. Traditional methods involve manual observation, which can be error-prone, especially with large datasets and subtle visual differences. An AI-driven solution is needed to streamline this process.

2. Proposed Solution:

•We propose Hematovision, a transfer learning-based solution using pre-trained convolutional neural networks such as ResNet50 and VGG16. These models will be fine-tuned on curated blood cell image datasets for effective and rapid classification.

3. Target Users:

•Medical professionals, pathologists, diagnostic laboratories, hospitals, AI researchers in medical diagnostics, and educational institutions for training purposes.

4. Expected Outcome:

• A user-friendly platform capable of processing microscopic images and classifying them into specific blood cell categories with high accuracy, aiding in the detection of infections, leukemia, and other hematological disorders.

Phase-2: Requirement Analysis

•Objective:

Conduct a thorough analysis of both technical and functional requirements. Identify all the software libraries, hardware capabilities, and computational needs essential for model training and deployment. Understand the clinical utility of each classification output and define the quality metrics required. Acknowledge ethical considerations and limitations such as potential misdiagnosis, data sensitivity, and patient privacy.

Key Points:

1. Technical Requirements:

•Languages: Python 3.10+
Frameworks: TensorFlow, Keras, OpenCV
Tools: Jupyter Notebook, Google Colab, VS Code
Hardware: GPU (NVIDIA RTX series recommended), 16GB RAM, SSD.

2. Functional Requirements:

1. Image upload interface
2. Classification of blood cells into categories (e.g., neutrophils, lymphocytes, eosinophils, monocytes)
3. Visualization of prediction confidence
4. Report download feature
5. API endpoints for external usage.

3. Constraints & Challenges:

1. Imbalanced datasets may bias model performance
2. Pre-trained models may require domain adaptation
3. High-resolution images demand more memory and processing
4. Interpretability and explainability for clinical use must be ensured

Phase-3: Project Design

•Objective:

Design a modular and scalable system architecture that clearly delineates data flow from input to output. Establish standards for user interaction, including how users provide inputs, interpret results, and receive feedback. Integrate UI/UX best practices for healthcare applications, ensuring that even non-technical users can navigate the system confidently and effectively.

•Key Points:

1. System Architecture Diagram:

•The system follows a modular architecture:
- Input module for image capture/upload
- Preprocessing module for standardizing input
- Inference module using transfer learning
- Output module with visualization and prediction report.

2. User Flow:

•User accesses the platform → Uploads or captures a microscopic image → Image is preprocessed (resized, normalized) → Model classifies cell type → Prediction with explanation (confidence and heatmaps) is displayed → Optionally export report.

3. UI/UX Considerations:

•Interface should be intuitive, mobile and desktop compatible. Use material design principles. Include accessibility features like font resizing and color-blind safe palettes. Error handling must be user-friendly with guided prompts.

Phase-4: Project Planning (Agile Methodologies)

- **Objective:**

Formulate a sprint-based project management plan allowing flexibility in development. Encourage team collaboration and frequent feedback loops. Track deliverables through a Kanban or Scrum framework. Use Agile principles to adapt based on model performance or usability test results, refining scope as necessary.

•Key Points:

1. Sprint Planning:

Sprint 0: Project setup & research
Sprint 1: Data acquisition and cleaning
Sprint 2: Baseline model setup

Sprint 3: UI/UX design
Sprint 4: Integration and testing
Sprint 5: Feedback and refinement.

2. Task Allocation:

Data Engineer: Image preprocessing, augmentation
- ML Engineer: Model training, validation
- UI Developer: Interface mockups and implementation
- Backend Developer: API, database integration
- QA Engineer: Test planning and automation.

3. Timeline & Milestones:

Week 1-2: Dataset finalized and preprocessed
Week 3-4: Model trained and tested on validation set
Week 5: UI connected to model
Week 6: End-to-end prototype and internal testing
Week 7-8: Deployment and documentation

Phase-5: Implementation

Objective:

Translate the design blueprint into a functional application. Ensure seamless integration of front-end interface, back-end APIs, and machine learning inference pipeline. Emphasize maintainable code, error handling, and reusability. Establish continuous testing and logging mechanisms to capture issues early in development.

Key Points:

1. Technology Stack Used:

Frontend: HTML5, CSS3, Bootstrap, JavaScript
Backend: Flask with RESTful APIs
Model: Keras with TensorFlow backend
Database (optional): Firebase/Firestore for logs
DevOps: GitHub, Heroku, Docker (if scaling).

2. Development Process:

1. Data Collection → 2. Preprocessing → 3. Model Design → 4. Training with transfer learning → 5. UI development → 6. Backend API creation → 7. Model Integration → 8. Testing & bug fixes.

3. Challenges & Fixes:

•Issue: Overfitting on smaller classes → Fix: Data augmentation, weighted loss functions
Issue: Slow inference time → Fix: Quantized model for deployment
Issue: Model size → Fix: MobileNetV2 alternative evaluated

Phase-6: Functional & Performance Testing

Objective:

Ensure that all functional requirements are met with high accuracy and consistency. Simulate real-world usage scenarios to validate stability, error handling, and UI behavior. Evaluate performance under resource constraints and peak usage. Verify if the model performs equally well across all expected blood cell types. Prepare for external validation by healthcare professionals.

Key Points:

1. Test Cases Executed:

•- Prediction Accuracy Tests
- Cross-validation with k-folds
- UI responsiveness and fail-safes
- Stress testing with large image batches
- Edge cases: blurred or low-res images

2. Bug Fixes & Improvements:

•Memory overflow issue during batch inference resolved with lazy loading. UI bug on image refresh fixed. Backend retry loop added for robustness.

3. Final Validation:

Model reached 94.6% test accuracy, 92% F1-score across five classes. Outperformed baseline CNN model. Reviewed and validated by medical consultant on a small curated test set.

4. Deployment (if applicable):

Deployed on Heroku for demo. Plans for AWS deployment with scalable GPU endpoint using SageMaker. Model containerized using Docker for local use or internal hospital networks.