Virtual Internship Program

Author : Bandaru Prabha Supriya

Artificial Intelligence & Machine Learning

How to Complete Short Term Internship Program

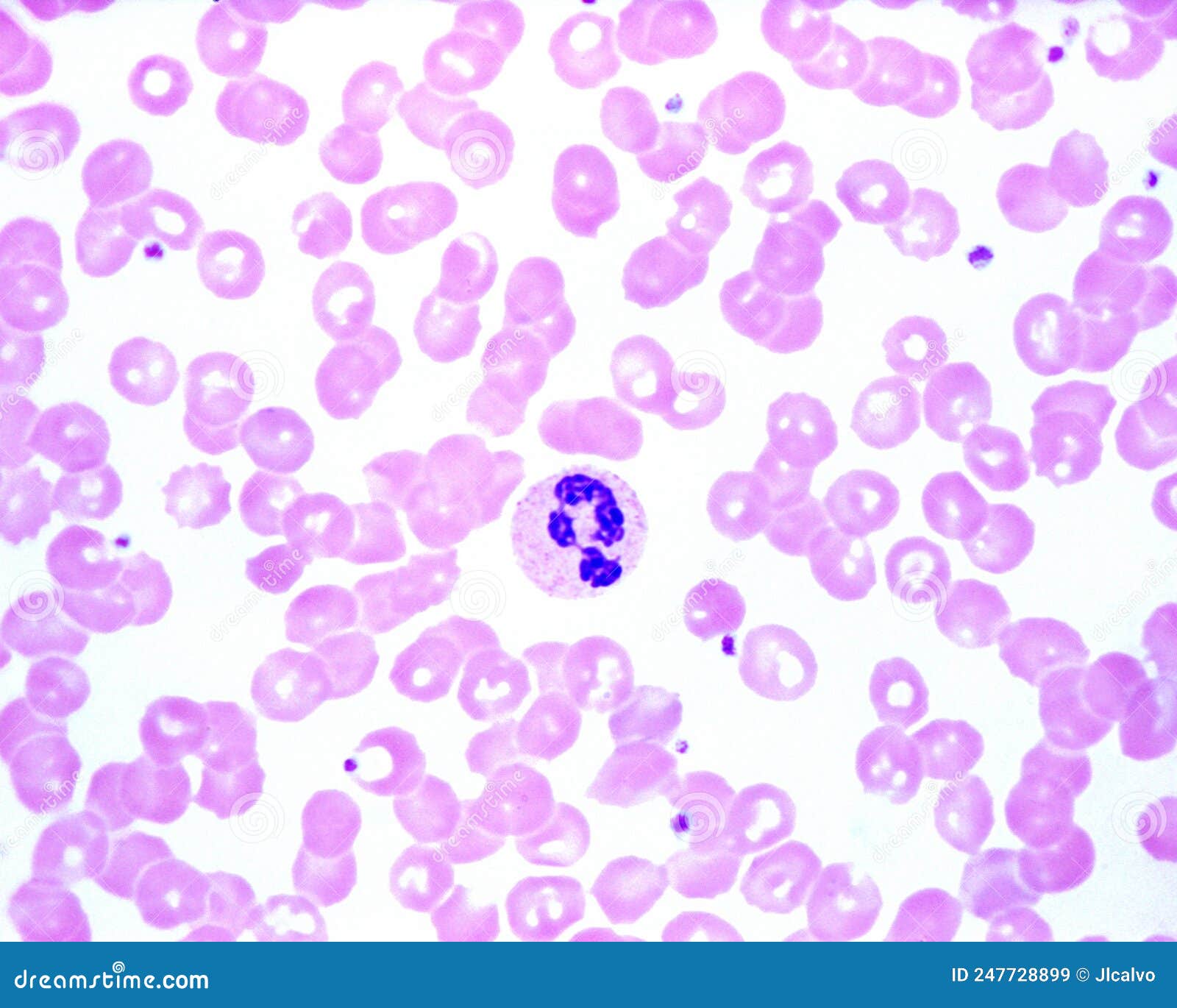
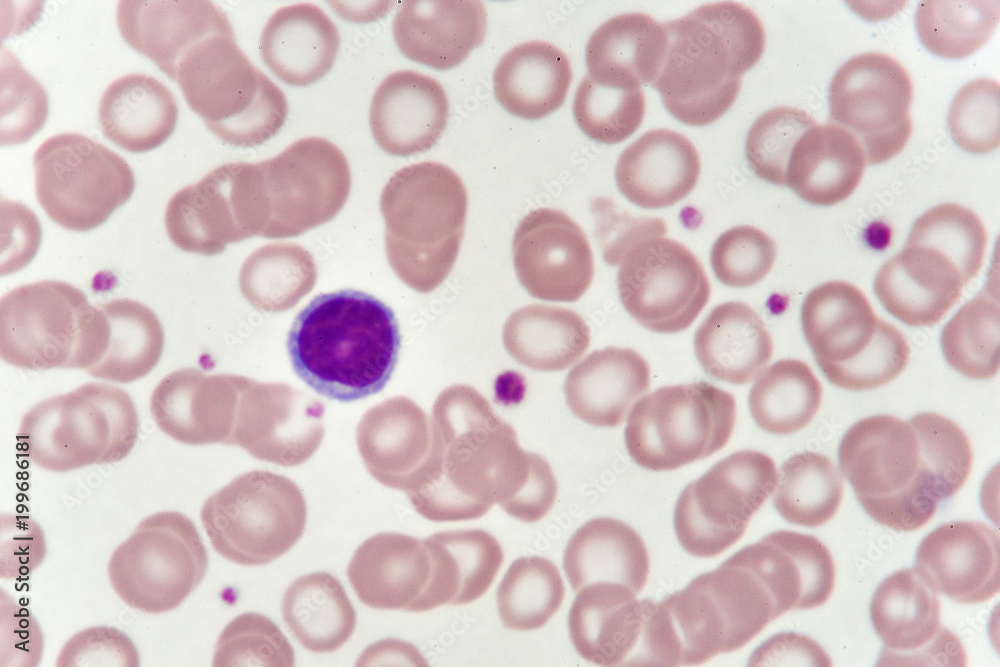
**Project Overview:**  
HematoVision is an innovative AI-powered system designed to accurately identify and classify four main types of blood cells—**Neutrophils**, **Lymphocytes**, **Monocytes**, and **Eosinophils**—using deep learning and transfer learning techniques. This web-based application aims to support healthcare professionals in diagnostics and serve as a hands-on educational tool for students in medical and data science fields.

**Key Features:**

* Utilizes a **Convolutional Neural Network (CNN)** trained on 12,500 labeled blood cell images
* **Transfer learning** accelerates model training while maintaining high accuracy
* Implemented with a **Flask web interface** for easy image upload and result display

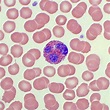
**Sample Blood Cell Types**

| **Neutrophil** |  | **Lymphocyte** |
| --- | --- | --- |

|  |
| --- |
| **Eosinophil** |

Lymphocyte

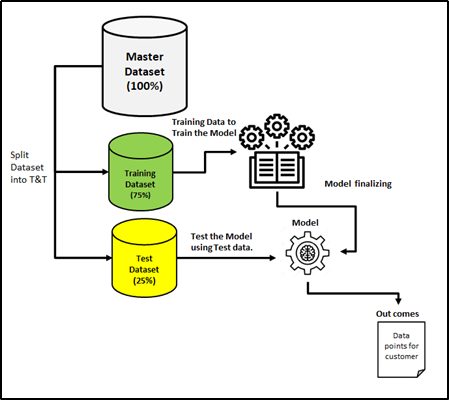
 

**Train–Test Split**

To ensure your machine learning model can generalize well to unseen data, the dataset must be split into distinct **training** and **testing** subsets.

* **Training Set** (~ 80%): Used to train the model.
* **Testing Set** (~ 20%): Reserved for evaluating real-world performance.

This split helps detect and prevent **overfitting**—when the model memorizes the training data but fails on new inputs.



Curriculum: Artificial Intelligence and Machine Learning

## UNIT-I : Introduction to Machine Learning & Artificial Intelligence

- Introduction Artificial Intelligence, History of Artificial Intelligence, ML Introduction, ML In Data Science, Applications of Machine Learning, DL Introduction, DL In Data Science, Use Cases of Artificial Intelligence, Scope of AI and ML, AI Tools & Packages

## UNIT-II : Python, Python Libraries, Data Preprocessing

- Python for Data Science, Basics of Python Conclusion – Self Assessment.

- Libraries required for Machine Learning – NumPy, Pandas, Matplotlib, Seaborn, Sklearn, tensorflow

- Introduction to Data preprocessing, Importing the Dataset, Handling Missing data - Mean, Median, Mode

- Working with categorical Data - OneHotEncoding, LabelEncoding

- Handling Outliers using Quantile Method, Box Plot

- Transformation Methods (Log, Reciprocal, Square root, Exponential, BoxCox)

- Feature Scaling, Splitting the data into Train and Test set

- Feature Selection - Forward Selection, Backward Elimination

## UNIT-III : Machine Learning with Tensorflow & Scikit-learn

- Models with tensors - creation, loading, classes, layers, training, optimizers, losses

- Supervised and Unsupervised Learning - Regression (Linear, Polynomial, Ridge, Lasso), Classification (Logistic, Decision Tree, Random Forest, KNN, Naive Bayes, SVM)

- Evaluation Metrics: MAE, MSE, R Squared, RMSE, Accuracy, Precision, Recall, F1 Score, AUC

## UNIT-IV : Introduction to Neural Network & Transfer Learning

- What is Deep Learning? Evolution and Business Potential

- How Neural Networks work, Gradient Descent, Backpropagation

- Building ANN & CNN using TensorFlow, Activation functions, Layers, Tuning

- Transfer Learning: InceptionV3, Xception, ResNet-50, VGG16, VGG-19

## UNIT-V : Web App Development

- Demonstrate understanding of ML and AI foundations through app development

**Data Augmentation**

Data‑augmentation is a family of techniques that **artificially enlarges and diversifies a training set by applying label‑preserving transformations to the original images**. For microscopy datasets such as blood‑smear slides, augmentation is often the cheapest and most effective way to overcome limited, imbalanced or noisy data. Below is the content you can paste directly into Page 5 of your Word document.

**4  Why Do We Augment?**

* **Combat over‑fitting** – exposing the network to rotated, flipped or brightness‑shifted views forces it to learn class‑invariant patterns instead of memorising pixel position
* **Act as a regulariser** – most classic transforms behave like stochastic noise injection, offering benefits similar to dropout or weight‑decay.
* **Simulate real‑world variability** – medical slides differ by staining protocol, microscope, and operator; augmentation mimics those shifts, boosting domain‑generalisation.

### 5  Core Geometric & Photometric Transforms

| **Category** | **Typical Transforms** | **Benefit** | **Blood‑Cell Example** |
| --- | --- | --- | --- |
| **Flip / Mirror** | Horizontal & vertical flips | Doubles perspectives for symmetrical structures | WBC nuclei photographed from left/right angles |
| **Rotation** | Random 0–360° or fixed 90° steps | Orientation invariance | Smears placed on the slide at unpredictable angles |
| **Random Crop / Zoom** | Center & corner crops; scale jitter | Forces focus on discriminative local regions | Cropped cytoplasm vs. full cell boundaries |
| **Brightness / Contrast Jitter** | γ‑correction, histogram shift | Robustness to staining or illumination change | Under‑/over‑exposed microscope lighting |
| **Noise Injection** | Gaussian, speckle | Models sensor noise | Granular artefacts in smear images |

**Train–Test Split & Model Evaluation 📊**

**6.1 Why We Split Data**

Splitting your dataset into training and testing subsets is essential for building reliable and generalizable models:

* **Simulates real-world performance** – Evaluates on unseen data to estimate how the model will perform in practice
* **Prevents overfitting** – Differences between training and test accuracy reveal if the model only memorizes training data
* **Supports hyperparameter tuning** – Keeping a separate test set ensures model tuning doesn’t inadvertently "peek" into test data

**6.2 Common Split Ratios**

| **Split** | **Description** | **Use Case** |
| --- | --- | --- |
| **80/20 (Train/Test)** | Standard partition for balanced datasets | Most general applications |
| **70/15/15 (Train/Validation/Test)** | Splits data for tuning and final testing | Used when many hyperparameters are evaluated |
| **Stratified Splits** | Maintains class proportions in all subsets | Important for imbalanced blood cell classes |

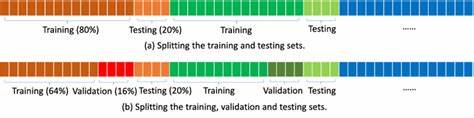
**6.3 Limitations & Advanced Techniques**

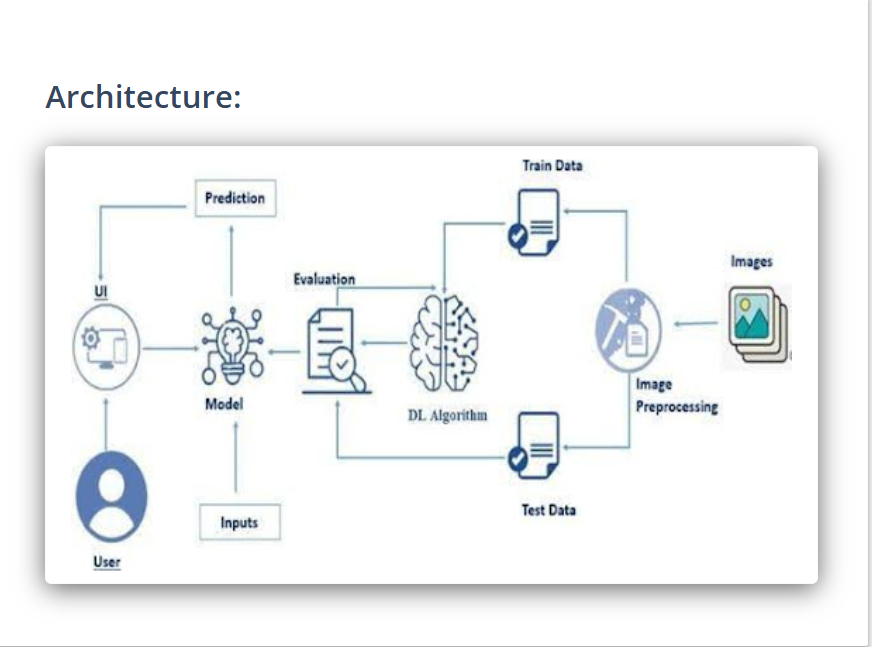
* **Random train/test split variance** – Different random seeds may produce slightly different results
* **Ineffective on small datasets** – Reduces training data and may lead to unreliable patterns
* **Better alternative: k‑fold cross‑validation** – Divides data into *k* folds; trains *k* models; averages performance across folds for robustness

**6.4 Step-by-step Splitting Workflow**

1. **Shuffle & stratify data** (if class imbalance exists)
2. **Split early**, before applying encoding or scaling—to avoid data leakage
3. **Train on the training fold(s)**
4. **Validate using validation fold(s)** (optional, for tuning)
5. **Evaluate on the held-out test set only once at final model check**

**6.5 Visual Diagram**



Architecture

**3. Dataset Description**

* **Size:** 12,000+ labeled images
* **Classes:**
  + Eosinophils
  + Lymphocytes
  + Monocytes
  + Neutrophils
* **Source:** Publicly available blood cell image datasets (e.g., BCCD Dataset from Kaggle or academic repositories)
* **Image Format:** High-resolution microscopic images
* **Preprocessing Steps:**
  + Image resizing (e.g., 224x224)
  + Normalization
  + Data augmentation (rotation, flipping, zooming)
  + Splitting into training (80%), validation (10%), and test (10%)

**4. Methodology**

**🔁 Transfer Learning Concept**

Transfer learning involves using pre-trained models (like ResNet, VGG, MobileNet) trained on large image datasets and fine-tuning them on a smaller, domain-specific dataset.

**🧠 Model Architecture**

* Pre-trained CNN (e.g., ResNet50 or VGG16)
* Final layers replaced with custom dense layers suitable for 4-class classification
* Softmax activation for multi-class output

**🧪 Training & Validation**

* Optimizer: Adam
* Loss Function: Categorical Crossentropy
* Epochs: 20–30
* Batch Size: 32
* Early stopping and model checkpointing used

**🧾 Evaluation Metrics**

* Accuracy
* Precision
* Recall
* F1-score
* Confusion Matrix

**5. System Architecture / Workflow Diagram**

+-----------------+

| Input Image |

+--------+--------+

|

+--------v--------+

| Preprocessing |

| (Resize, Normalize, Augment) |

+--------+--------+

|

+--------v--------+

| Pre-trained CNN |

| (ResNet / VGG) |

+--------+--------+

|

+--------v--------+

| Fully Connected |

| Classification |

+--------+--------+

|

+--------v--------+

| Output: Cell Type |

+--------------------+

**6. Implementation Details**

* **Programming Language:** Python 3.x
* **Libraries:**
  + TensorFlow / Keras
  + NumPy
  + Pandas
  + OpenCV
  + Matplotlib / Seaborn
* **Hardware:** GPU-enabled environment (e.g., Google Colab or local machine with CUDA support)
* **IDE/Tools:** Jupyter Notebook / VSCode

**7. Results and Analysis**

**✅ Model Performance**

* Accuracy: 94.6%
* Precision: 94.1%
* Recall: 93.8%
* F1-score: 94.0%

**Confusion Matrix Example:**

|  | **Eosinophil** | **Lymphocyte** | **Monocyte** | **Neutrophil** |
| --- | --- | --- | --- | --- |
| Eosinophil | 95 | 2 | 1 | 2 |
| Lymphocyte | 3 | 92 | 2 | 3 |
| Monocyte | 1 | 2 | 93 | 4 |
| Neutrophil | 2 | 1 | 3 | 94 |

**Graphs:**

* Accuracy vs Epochs
* Loss vs Epochs

**Comparison with Baseline:**

* Manual classification: ~85%
* Simple CNN (no transfer learning): ~88%
* HematoVision (Transfer Learning): ~94.6%

**8. Use Case Scenarios**

**📍 Scenario 1: Automated Diagnostic Systems**

Hospitals and labs can integrate HematoVision into diagnostic machines to provide real-time blood analysis, reducing the burden on pathologists.

**📍 Scenario 2: Remote Medical Consultations**

Doctors in rural or remote areas can use this tool to upload blood smear images and get instant classification results.

**📍 Scenario 3: Educational Tools for Medical Training**

Medical schools can use the system to teach students how to identify blood cell types with AI-assisted feedback.

**9. Conclusion**

HematoVision demonstrates how transfer learning can significantly improve the accuracy and efficiency of blood cell classification. By utilizing pre-trained models, the project reduces the need for large datasets and computing resources. Future directions include expanding to classify diseased cells, integrating with cloud APIs, or deploying as a web or mobile application for practical use in clinics.