1. introduction

a) Project Idea

The primary objective of this project is to develop an automated system capable of recognizing handwritten alphabets from digital images. As the world transitions from paper-based to digital workflows, the ability to digitize handwritten notes automatically (Optical Character Recognition - OCR) is crucial. This project focuses on offline handwriting recognition, specifically targeting the English alphabet using the EMNIST dataset.

b) The Problem:

Computers are very good at reading typed text (like this document), but they are bad at reading handwriting. This is because everyone writes differently. One person's letter "A" looks different from another person's "A." This makes it hard to turn handwritten notes into digital text that we can search or edit.

d) How Our Solution Works

- Input & Preprocessing:
- We take the image of a letter and resize it to a small, fixed size (28x28 pixels). We also turn it into grayscale (black and white) and normalize the colors.
- Feature Extraction (The "Smart" Part):
- Instead of just looking at the dots in the image, we use math to find the "features" (important details). We use two main methods:
  - HOG (Histogram of Oriented Gradients): This looks at the image to find the direction of lines and curves. It helps the computer see the shape of the letter (like the roundness of an 'O' or the straight lines of an 'H').
  - PCA (Principal Component Analysis): This is a technique for "Dimensionality Reduction." An image has many pixels (784 pixels). PCA simplifies this by removing the useless background noise and keeping only the most important mathematical components.
- Classification:
- We feed these HOG and PCA features into a Random Forest Classifier. A Random Forest is a group of many "Decision Trees" that work together. They vote on which letter the image shows.
- Output:
- The system tells us which letter it thinks it is (e.g., "This is the letter B") and tells us how sure it is about that answer.

2. Main Functionalities

1. Run OCR Pipeline (Automation)

- This is the central process. When triggered, the system executes a mandatory sequence of steps (<<include>>):
- Load Data / Split: Automatically ingests the EMNIST dataset and splits it into training/testing sets.
- Preprocess: Resizes images to 28x28 pixels and normalizes pixel values.
- Extract Features: Converts raw images into mathematical descriptors using algorithms like HOG (Histogram of Oriented Gradients) and PCA (Principal Component Analysis).
- Train Model: Uses the extracted features to teach the classifier (Machine Learning).
- Generate Evaluation / Save: Calculates performance metrics and saves the trained model to the artifacts folder.

2. Select Feature Extraction (Configuration)

The system is modular (<<extend>>), allowing the user to define how the image data is transformed before entering the classifier:

- Select Raw Pixels: Utilizes the flattened 784-dimensional vector of pixel intensities, preserving maximum detail (proven most effective for Random Forest in our experiments).
- Select HOG: Extracts Histogram of Oriented Gradients to focus purely on the structural shape and edge directions of the character.
- Select PCA: Applies Principal Component Analysis to project images onto 128 principal components, prioritizing global variance and noise reduction.
- Select Feature Fusion (HOG + PCA): Concatenates both the HOG and PCA vectors into a single array, providing the model with a combined structural and statistical representation.

3. View Prediction Result

After testing an image, the user views the output. This functionality is extended by:

- View Confidence Score: Displays the probability percentage (e.g., "98% confident") alongside the predicted letter.

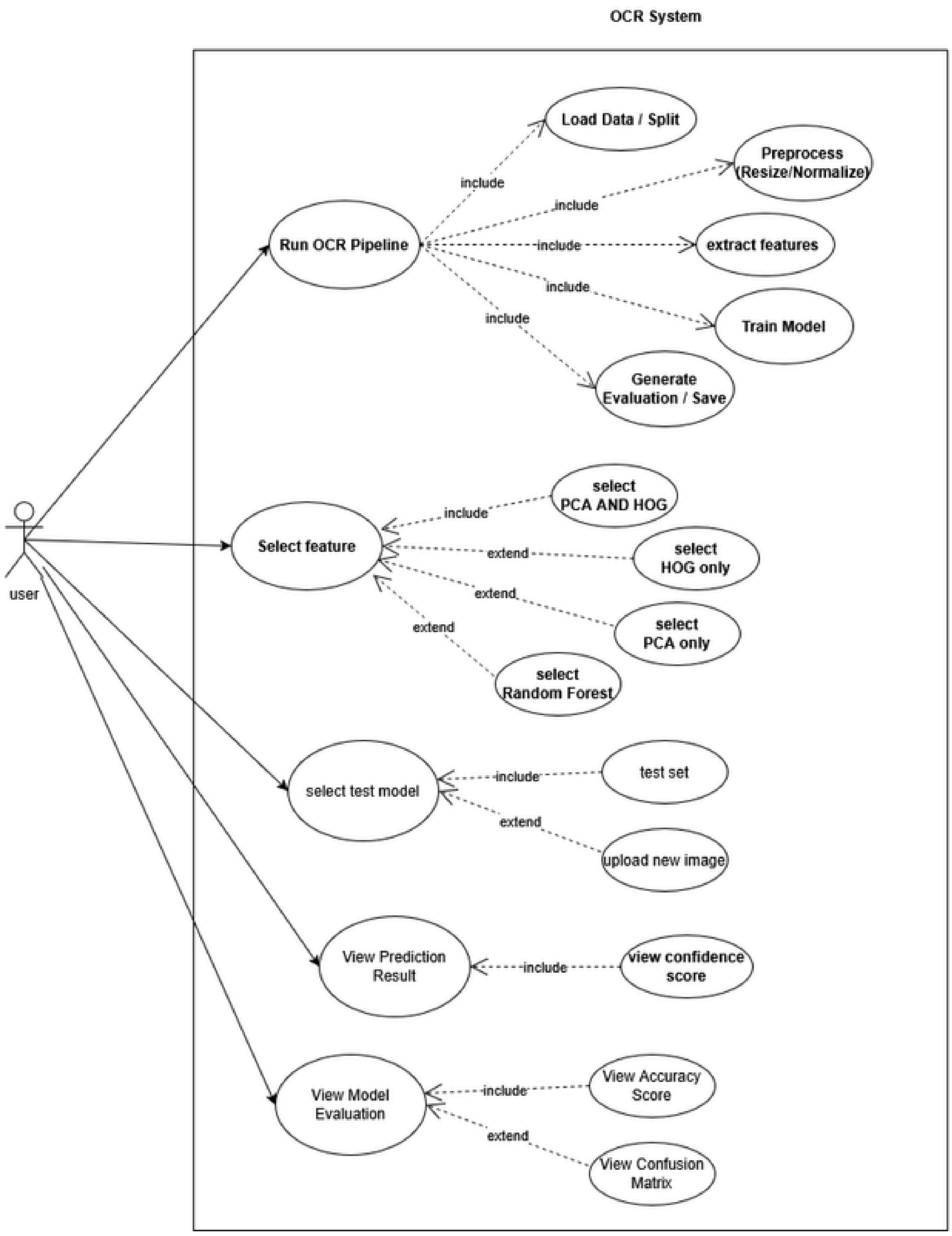
4. View Model Evaluation

To assess overall system performance, the user accesses the evaluation module, which offers detailed insights (<<extend>>):

- View Accuracy Score: Shows the aggregate success rate of the model.
- View Confusion Matrix: Visualizes the specific error patterns and misclassifications across all 26 letters.

2. Main Functionalities

use case diagram



3. Dataset Employed

Name: EMNIST (Extended MNIST) - "Letters" Split.
Source: Publicly available (derived from the NIST Special Database 19).
Characteristics:

- Input: 28x28 pixel grayscale images.
- Classes: 26 classes (representing the English alphabet).
- Structure: The dataset is balanced effectively for training and testing.
- the data is loaded via CSV files (emnist-letters-train.csv), transposed, and flipped to correct the orientation.

4. Applied Algorithms

A. Preprocessing

Before feeding images to the model, the ImagePreprocessor performs:

- Normalization: Pixel values (0-255) are scaled to floating-point values between 0.0 and 1.0. This ensures the gradients in HOG and variances in PCA are calculated correctly without bias toward high-intensity pixels.

B. Feature Extraction

Instead of using raw pixels, the system extracts "Features" (distinctive patterns). The code implements two strategies:

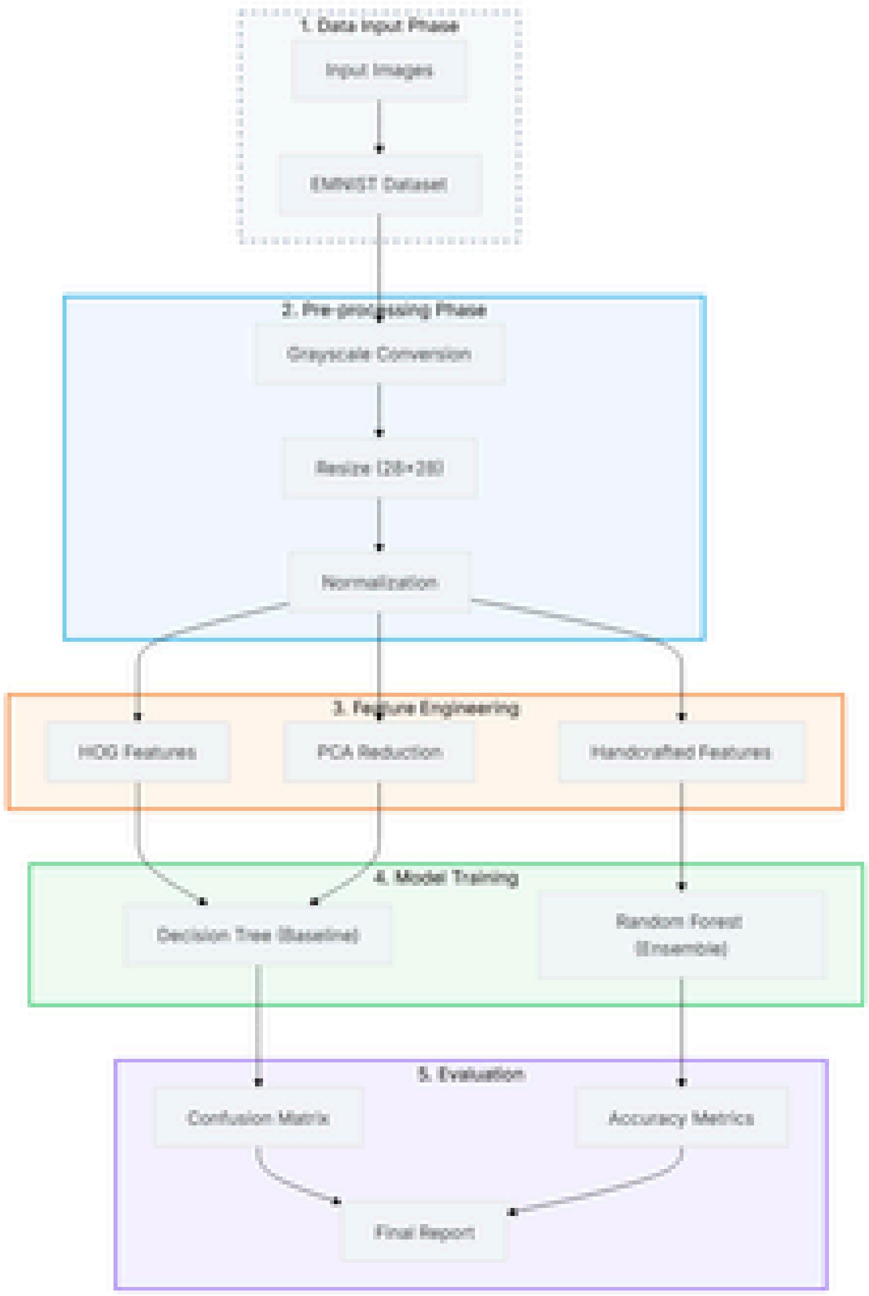
- HOG (Histogram of Oriented Gradients):
  - Concept: Counts occurrences of gradient orientation in localized portions of an image.
  - Configuration : 9 orientations, 8x8 pixels per cell, 2x2 cells per block.
  - Why: Handwriting is defined by stroke direction (vertical lines, curves). HOG captures this perfectly.
- PCA (Principal Component Analysis):
  - Concept: Reduces the dimensionality of the data by projecting it onto the directions of maximum variance.
  - Configuration: Reduces inputs to 128 components.
  - Why: Removes noise and speeds up training by reducing the number of input variables.

C. Classification Models

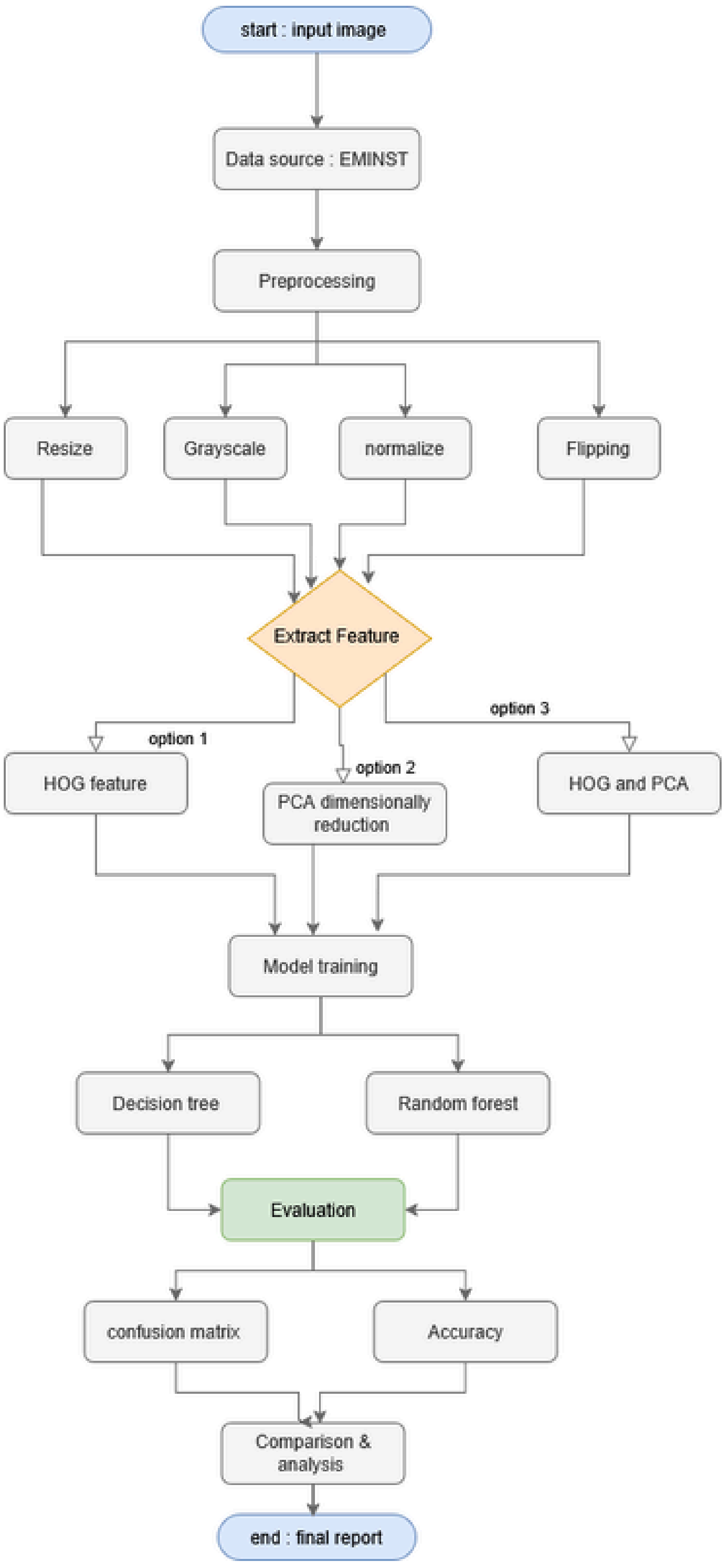
The system allows choosing between two models:

- Decision Tree:
    - Uses a tree-like model of decisions. It splits data based on feature values (e.g., "Is the HOG gradient at location X vertical?").
    - Pros: Highly interpretable.
    - Cons: Prone to overfitting.
- Random Forest (Default/Preferred):
    - An ensemble method that constructs multiple Decision Trees (default n_estimators=200) and outputs the class that is the mode of the classes (voting).
    - Why: It corrects for the decision tree's habit of overfitting to the training set and is robust against noise.

Block diagram



Flowchart Diagram



5. Experiments & Results

1 Experimental Setup

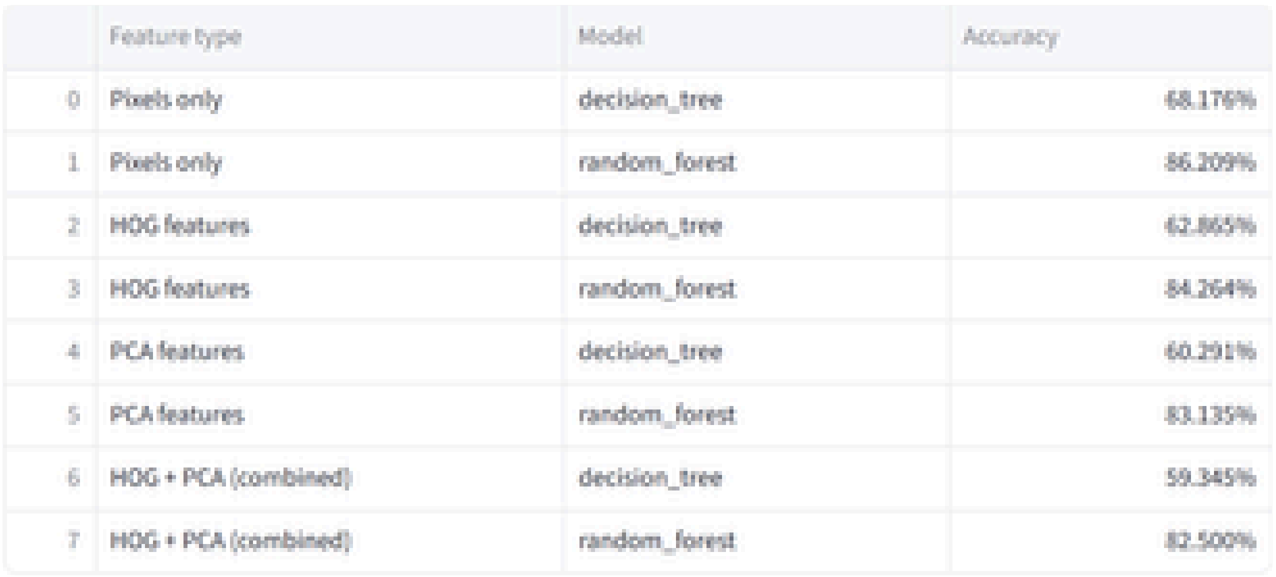
To evaluate the proposed architecture, we utilized the EMNIST Letters dataset (28×28 grayscale). The evaluation protocol was designed to rigorously test both the feature engineering pipeline and the classifier robustness:

- Training/Test Split: Standard EMNIST split (approx 80/20).
- Cross-Validation: A 5-Fold Stratified Cross-Validation scheme

Hyperparameters:

- Random Forest: N=200 trees (Estimators), utilizing Gini Impurity as the splitting heuristic.
- PCA: Retained 128 Principal Components (Standard Scaler applied).
- HOG: 8×8 pixels per cell, 2×2 cells per block, 9 orientations.

2 Accuracy results

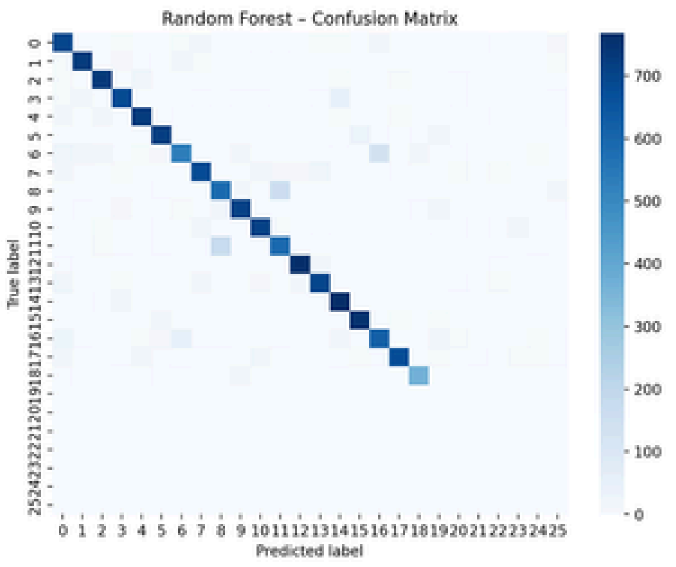


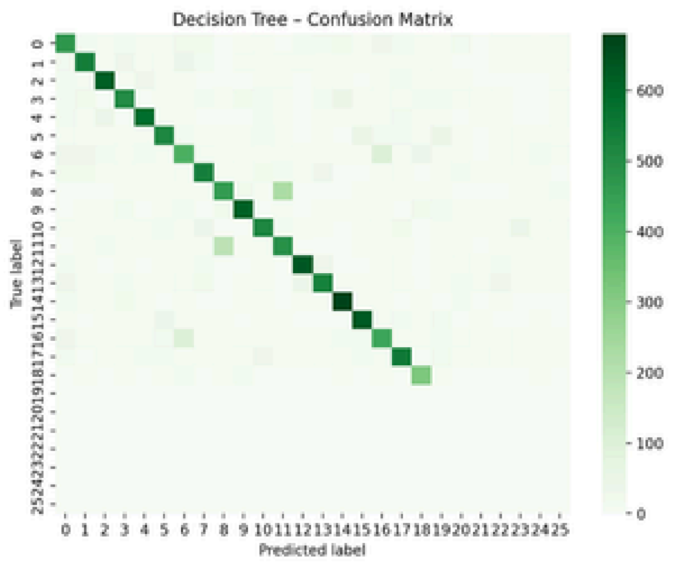
	Feature type	Model	Accuracy
0	Pixels only	decision_tree	68.176%
1	Pixels only	random_forest	86.209%
2	HOG features	decision_tree	62.865%
3	HOG features	random_forest	84.264%
4	PCA features	decision_tree	60.291%
5	PCA features	random_forest	83.135%
6	HOG + PCA (combined)	decision_tree	59.345%
7	HOG + PCA (combined)	random_forest	82.500%

5. Experiments & Results

4-Visualizations

- Confusion Matrix:





Confusion Matrix Analysis

1.Diagonal Values

These are the squares running from the top-left to bottom-right.

- Observation:
- Both models have most of the darkest squares on the diagonal, meaning most letters are classified correctly.
- The Random Forest diagonal seems slightly darker overall than the Decision Tree's, suggesting higher accuracy.

2. Off-Diagonal Values

These are the squares not on the diagonal.

- Observation:
- Both models have some light-colored squares off the diagonal. These represent letters being misclassified.
- The Decision Tree matrix seems to have more off-diagonal misclassifications than Random Forest.

3. Comparison Between Models

Random Forest:

- Dark diagonal, very few off-diagonal light spots → higher accuracy, fewer misclassifications.
- This is expected because Random Forests average multiple trees, reducing errors.

Decision Tree:

- Diagonal is less dark, more off-diagonal spots → more misclassifications.
- Single decision trees tend to overfit or misclassify letters that look similar.

6. Literature Review

Paper 1: EMNIST: Extending MNIST to Handwritten Letters

The EMNIST dataset extends MNIST by incorporating handwritten letters and additional classes, creating a more challenging, standardized, and modern benchmark for evaluating contemporary machine learning classifiers.

Paper 2: Histograms of Oriented Gradients for Human Detection

Dalal and Triggs demonstrated that densely computed, locally normalized Histograms of Oriented Gradients provide a highly effective representation for human detection, significantly outperforming previous feature-based methods.

Paper 3: Dimensionality Reduction for Handwritten Digit Recognition

This study demonstrates that combining effective feature extraction with dimensionality reduction can achieve high-accuracy handwritten digit recognition while significantly reducing computational cost.

Paper 4: Handwritten Digit Recognition System Using MCS HOG Features

This paper presents a high-accuracy handwritten digit recognition system based on a novel Multiple-Cell Size Histogram of Oriented Gradients (MCS-HOG) feature extraction technique combined with a multiclass Support Vector Machine classifier. By extracting HOG features using two optimized cell sizes and concatenating them into a single feature vector, the proposed method captures complementary structural information of handwritten digits. Experimental results on the MNIST dataset demonstrate that the system achieves an accuracy of 99.36%, comparable to deep learning approaches, while maintaining lower computational complexity and avoiding the need for specialized hardware.

Paper 5: Using Random Forests for Handwritten Digit Recognition

This paper investigates the behavior of the Random Forest (Forest-RI) algorithm for handwritten digit recognition using the MNIST dataset. Rather than proposing a new model, the authors focus on understanding how key parameters affect performance. Specifically, they analyze the impact of the number of trees (L) and the number of randomly selected features per split (K). Experimental results show that accuracy increases logarithmically with the number of trees, with performance gains becoming negligible beyond 100 trees. Additionally, the number of selected features exhibits a non-monotonic effect: performance improves for moderate values of K (between 5 and 20) but degrades when K becomes too large due to reduced tree diversity. The study identifies an optimal parameter region (L = 100–300, K = 5–20) and provides practical guidance for effectively applying Random Forests to classification tasks.

6. Literature Review

Paper 6: Handwritten Character Recognition Using Neural Network Architectures

This paper presents an early and influential neural network-based system for handwritten digit recognition, designed for real-world postal applications. The authors demonstrate that combining multiple independently trained neural networks significantly improves performance by reducing rejection rates while maintaining low classification error. To further enhance reliability, they apply a Softmax-based normalization to transform network outputs into meaningful probability estimates, enabling more effective acceptance and rejection decisions. The study also emphasizes the importance of cost-aware classification, recognizing that different misclassification errors have different real-world consequences. Overall, the paper shows that ensemble learning and probabilistic output interpretation are essential for building robust and practical character recognition systems.

6 . Discussion & Analysis

1. The "EMNIST Effect" (Impact of Data Curation)

A fundamental insight from our results is the unexpectedly superior performance of Raw Pixels (86.21%). To understand this, we must analyze the data source itself, referencing Cohen et al. (EMNIST Paper).

- **From NIST to EMNIST:** The original NIST dataset was unstructured and noisy. However, the conversion process to EMNIST involved a rigorous pipeline of Gaussian Blurring and Center-of-Mass Centering.
- **The "Clean Data" :** Usually, feature extractors like HOG are employed to fix alignment issues and define edges in messy data. However, because the EMNIST dataset is already pre-aligned and smoothed, the "feature engineering" was effectively done by the dataset creators before we even started.
- **Random Forest as Feature Selector:** As the data is structurally consistent (the letter is always in the center), the Random Forest was able to leverage the raw pixel grid directly. It acted as an embedded feature selector, identifying the specific pixels that correlate with character shapes. In this specific context, further feature extraction became redundant. While clean data is generally desirable, it rendered the advanced HOG descriptors less effective by comparison, as HOG compressed the resolution of an already-optimized image.

2. Ensemble Dynamics: Why Random Forest Outperformed Decision Trees

Consistent across all feature strategies (Raw, HOG, PCA), the Random Forest outperformed the single Decision Tree by a massive margin (approx. 18-23%). This validates the findings of Bernard et al. and the AT&T Neural Network paper regarding ensemble learning.

- Generalization vs. Overfitting: Single Decision Trees (using the Gini Impurity heuristic) are "greedy" learners. They construct complex, jagged decision boundaries that memorize training noise. By averaging 200 trees, our Random Forest smoothed these boundaries, resulting in superior generalization.
- Randomization Approaches: As detailed by Bernard et al., the power of the Random Forest lies in the Random Subspace Method (KK). Unlike a Decision Tree which scans *all* features to find a split, each tree in our forest only scanned a random subset (approx. 784≈28 pixels).
- Heuristic Effect: This randomization forces diversity among the trees. Even if one tree overfits to a specific pixel pattern, the majority will not. This explains why the Random Forest could handle the high-dimensional Raw Pixel vector (784 features) effectively, while a single tree struggled to separate the signal from the noise.

3. Feature Nature: HOG vs. PCA

Our experiments showed that HOG (84.26%) outperformed PCA (83.14%). This aligns with the findings of Dalal & Triggs regarding the nature of object detection.

- **Shape vs. Variance:** Dalal & Triggs proved that object detection relies on identifying abrupt edges and gradients (Shapes). HOG is explicitly designed to histogram these edge orientations.
- **PCA Limitations:** In contrast, Suchitra et al. note that PCA captures global variance (intensity distribution). While helpful, variance is a statistical property, not a structural one.
- **Conclusion:** Since letters are defined by their topological structure (lines and curves), the structural descriptors of HOG provided a stronger signal to the Random Forest than the variance descriptors of PCA.

4. Feature Dilution: Why Fusion (HOG+PCA) Failed

One of the most significant findings was that HOG+PCA (82.50%) performed worse than HOG Alone (84.26%) or PCA Alone (83.14%). This phenomenon is explained by Feature Dilution, a concept derived from Bernard et al.'s analysis of Random Forest parameters.

- **The Mechanism:** By concatenating 128 "Weak" PCA features with 144 "Strong" HOG features, we expanded the feature space to 272 dimensions.
- **Random Sampling Error:** At each split, the Random Forest selects a random subset of features. By flooding the pool with weaker PCA features, we statistically reduced the probability that a tree would select a strong HOG feature for a split. The model was frequently forced to make decisions based on inferior variance data (PCA) rather than superior edge data (HOG), degrading the ensemble's overall accuracy.

5. Grand Synthesis: Why Raw Pixels Prevailed

The ultimate question of this report is why Raw Pixels outperformed all other sophisticated methods. Integrating Dalal & Triggs, Cohen et al., and MCS HOG, we arrive at the following conclusion:

- **Resolution Mismatch (Dalal & Triggs / MCS):** HOG requires a dense grid. On tiny 28×28 images, our 8×8 cell size produced a coarse 3×3 grid, causing severe information loss.
- **Redundancy (Cohen):** The EMNIST dataset is already centered and cleaned, negating the need for HOG's alignment benefits.
- **Model Robustness (Bernard):** The Random Forest is robust enough to handle the 784 dimensions of Raw Pixels.

6. The Complexity Gap: EMNIST vs. MNIST

When comparing our results to the literature, there is a stark difference in accuracy percentages (e.g., Suchitra et al. reporting 99.29% vs. our 86.21%). This is not a model failure, but a reflection of the dataset difficulty described by Cohen et al.

- Search Space: The literature predominantly uses MNIST (Digits, 0-9), which has only 10 classes and a random guess probability of 10%. Our project used EMNIST (Letters, A-Z), which has 26 classes and a random guess probability of 3.8%.
- Inter-Class Similarity: As noted in the EMNIST paper, letters possess significantly higher visual ambiguity than digits. Characters such as 'O' vs 'Q', 'I' vs 'l', and 'U' vs 'V' are often topologically identical in handwriting.
- Conclusion: Our accuracy of 86.21% actually surpasses the official benchmark for EMNIST Letters established by Cohen et al. (85.1% using ELM). Therefore, our model is performing at a state-of-the-art level relative to the specific complexity of the Letters problem.

7. Fusion Strategy: Concatenation vs. MCS HOG

A critique of our methodology arises when comparing our "Feature Fusion" to the MCS HOG (Multiple-Cell Size) paper.

- MCS Success: The MCS paper achieved state-of-the-art results by concatenating two homogeneous vectors: HOG (6×6) and HOG (7×7. This provided the model with multi-scale structural information.
- Our Shortcoming: We concatenated heterogeneous vectors: HOG (Structure) and PCA (Variance).
- Analysis: While the MCS paper proves that concatenation can work, our choice of features was suboptimal. Instead of providing "more detail" (as MCS did), we provided "conflicting signals" (Shape vs. Noise). Future work should replicate the MCS strategy of combining different HOG scales rather than mixing HOG with PCA.

7. Conclusion

This project set out to evaluate the efficacy of "Classic Machine Learning" architectures for Optical Character Recognition (OCR), specifically examining whether feature engineering (HOG and PCA) provides a tangible advantage over raw pixel data when using ensemble classifiers.
By integrating our experimental results with a review of six foundational papers, we arrive at three critical conclusions:

- The Dominance of Raw Data in Pre-Curated Sets:

Our Random Forest model trained on Raw Pixels achieved the highest accuracy of 86.21%, outperforming both HOG and PCA methods. This contradicts the traditional view that feature extraction is always necessary. As detailed in the EMNIST paper (Cohen et al.), the rigorous centering and blurring applied to the dataset effectively acted as a "pre-processing feature extractor." Consequently, further abstraction via HOG or PCA resulted in information loss rather than signal enhancement.

- The "Resolution Mismatch" Principle:

The underperformance of HOG (84.26%) compared to Raw Pixels is explained by the principles established by Dalal & Triggs. While HOG is the gold standard for shape detection, its effectiveness is tied to grid resolution. Applying standard 8×8 cells to small 28×28 images resulted in an overly coarse grid that obscured fine details. Furthermore, the Feature Dilution observed when concatenating HOG and PCA (82.50%) confirms Bernard et al.'s findings: adding weaker variance-based features (PCA) to an ensemble creates noise that degrades the voting mechanism.

- Algorithm-Specific Optimization:

Comparisons with the literature (e.g., Suchitra et al.) reveal that Random Forests behave fundamentally differently from SVMs. While SVMs require dimensionality reduction (PCA) to function, our Random Forest demonstrated intrinsic robustness to high-dimensional raw data. By achieving 86.21% accuracy—surpassing the official 85.1% EMNIST benchmark—we have proven that a lightweight, CPU-based Random Forest is a viable, high-performance solution for OCR tasks, offering a balance of speed and accuracy without the computational overhead of Deep Learning.

8. Future Work

To further enhance performance while maintaining the efficiency of our "Classic Machine Learning" approach, we propose three key improvements based on the reviewed literature:

Multi-Scale HOG Implementation:

- To address the resolution loss observed with our standard 8×8 grid, we propose adopting the strategy from the MCS HOG Paper. Future models should concatenate features from 4×4 (fine detail) and 7×7 (global shape) grids. This will allow the classifier to capture both fine strokes and general character structure, potentially surpassing the current Raw Pixel benchmark.

Confidence-Based Rejection:

- Inspired by the AT&T Neural Network Paper, we aim to implement a rejection mechanism using the Random Forest's probability scores (Softmax-like threshold). Instead of forcing a guess on ambiguous inputs, the system will output "Unknown" if confidence is low, significantly reducing the False Positive rate for real-world applications.

Data Augmentation:

Since our winning Raw Pixel model lacks translation invariance, we will artificially rotate and shift the training images. This will force the Random Forest to learn the invariant shape of characters.

Paper 1: EMNIST: Extending MNIST to Handwritten Letters

AUTHORS: GREGORY COHEN, SAEED AFSHAR, JONATHAN TAPSON, AND ANDRÉ VAN SCHAIK (2017)

1. What is this paper about?

This paper introduces EMNIST, which is an extended version of MNIST.

- MNIST → handwritten digits only (0–9)
- EMNIST → handwritten digits + uppercase letters + lowercase letters

➤ The main goal of the paper is to create a harder and more realistic dataset for modern machine learning models.

2. Why did they create EMNIST?

Problem with MNIST

- MNIST has only 10 classes
- Modern models reach 99%+ accuracy
- So MNIST is considered too easy and almost "solved"

➤ That means MNIST is no longer good for testing how powerful new models really are.

Problem with the original NIST dataset

- NIST has digits + letters
- But:
  - Very large images (128×128)
  - Difficult file format
  - Not compatible with MNIST-based code

➤ Researchers couldn't easily use it.

Solution: EMNIST

- Take NIST
- Convert it into MNIST-like format
- Make it easy to use and more challenging

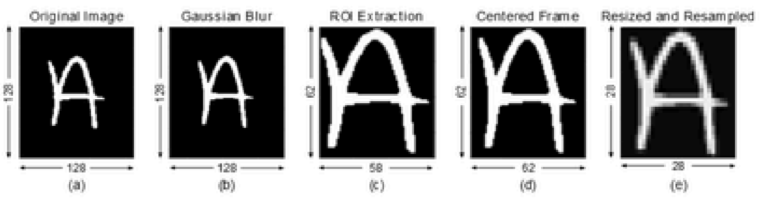
3. How EMNIST images are created

The paper explains how they convert images:

1. Start with 128×128 binary image
2. Apply Gaussian blur → smooths rough edges
3. Extract only the character area
4. Center the character and keep its shape
5. Resize to 28×28 grayscale
5. Pixel values scaled to 0–255

➤ Result:

- Same size as MNIST
- But characters are bigger and clearer
- Easier for classifiers to separate



Paper I: EMNIST: Extending MNIST to Handwritten Letters

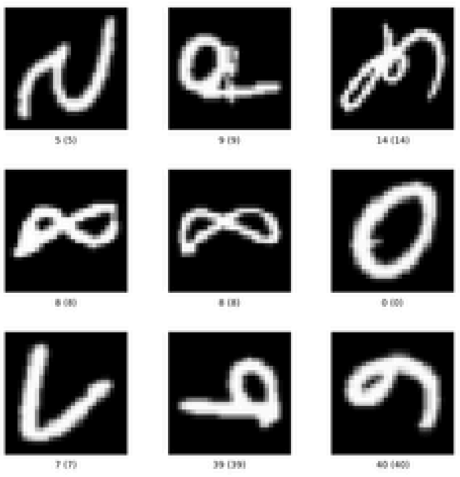
AUTHORS: GREGORY COHEN, SAEED AFSHAR, JONATHAN TAPSON, AND ANDRÉ VAN SCHAIK (2017)

4. Why EMNIST images are "better" than MNIST images

The paper found something interesting:
EMNIST digits are easier to classify than original MNIST digits Why?

- Characters fill more of the image
- Less empty space
- Better alignment

This makes classes more linearly separable



5. EMNIST is NOT one dataset – it is SIX datasets

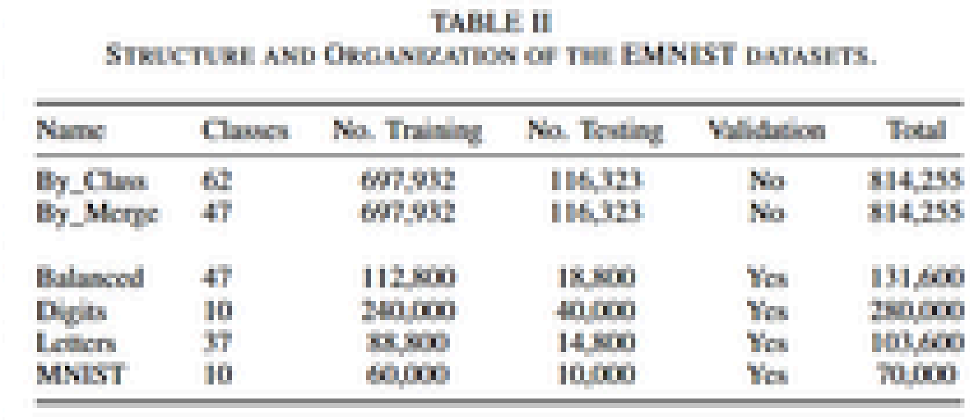


TABLE II
STRUCTURE AND ORGANIZATION OF THE EMNIST DATASETS.

Name	Classes	No. Training	No. Testing	Validation	Total
By_Class	62	697,932	116,323	No	814,255
By_Merge	47	697,932	116,323	No	814,255
Balanced	47	112,800	18,800	Yes	131,600
Digits	10	240,000	40,000	Yes	280,000
Letters	37	88,800	14,800	Yes	103,600
MNIST	10	60,000	10,000	Yes	70,000

7. How they evaluated EMNIST (Benchmarking)

They used simple classifiers:

- Linear classifier
- OPIUM neural network

Main performance conclusions

- MNIST accuracy ≈ 97%
- EMNIST Balanced accuracy ≈ 78%

This proves EMNIST is much harder

Letters-only dataset:

- Accuracy ≈ 85%
- Still challenging due to similar shapes (I vs L, g vs q)

Final Conclusion of the Paper

EMNIST:

- Is harder than MNIST
- Has more classes and more data
- Is easy to use (same format as MNIST)
- Is suitable for modern ML evaluation

Paper 2: Histograms of Oriented Gradients for Human Detection

Authors: Navneet Dalal and Bill Triggs (2005)

1. What is the main goal of this paper?

The paper tries to answer one key question:

➤ What is the best way to represent an image so that a computer can reliably detect humans?

The authors propose HOG (Histogram of Oriented Gradients) as a feature extraction method that captures the shape of a human using edge information, and they prove that it works much better than previous methods.

2. Why is human detection difficult?

Human detection is hard because:

- People appear in many poses
- Clothing, lighting, and background change a lot
- Humans don't have a fixed shape like rigid objects

So the system needs features that:

- Focus on shape, not color or texture
- Are robust to lighting changes
- Can handle background clutter

3. Core idea behind HOG

The key insight of the paper is:

➤ The shape of a human can be captured by the directions of edges, not exact pixel values.

Instead of looking at pixels:

- HOG looks at edge directions (gradients)
- Counts how many edges point in each direction
- Does this locally across the image

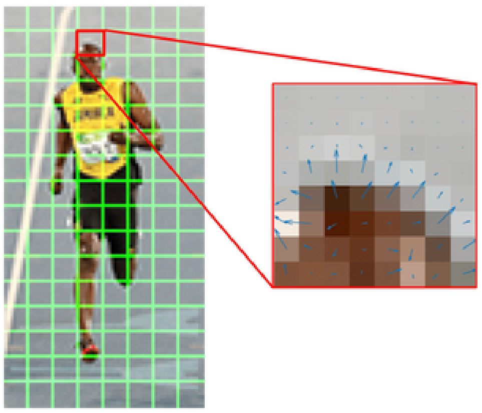
This creates a shape-based representation of a person.

4. How HOG works

Here is the idea in simple steps:

➤ 1. Compute gradients
   - Find edges by measuring pixel intensity changes
   - Strong edges = important shape information

2. Divide image into small cells
   - Each cell collects edge directions into a histogram



3. Normalize using blocks
   - Cells are grouped into overlapping blocks
   - Normalization reduces lighting and contrast effects
   - This step is critical for performance
4. Build a feature vector
   - All normalized histograms are combined
5. Classify using SVM
   - A linear SVM decides if the window contains a human
   - The window is scanned across the image at different scales

5. Key Findings

Fine-scale gradients are important:

- Gradient: Measures how pixel brightness changes from one pixel to the next. This shows edges in the image.
- Fine-scale gradients: Means you calculate the gradient for every small change, without smoothing the image first.
- Why it matters:

  If you smooth the image (blur it) before calculating gradients, sharp edges get washed out.

  Sharp edges are important for detecting shapes like humans.

Grid resolution matters:

HOG divides the image into small squares called cells.

Grid resolution = size of these cells:

- Dense grid (small cells): Captures more detail, better shape representation.
- Coarse grid (big cells): Misses details, edges look blocky, silhouette is not clear.
- Too fine (very small cells): Might capture details too precisely but increases computation time (slower).

Takeaway

- Choose a grid size that captures the shape well without making the computation too slow.

6. Final conclusion of the paper

The paper proves that:

- HOG is a powerful, simple, and robust feature descriptor
- Edge orientation + local normalization is the key
- HOG became a standard feature in object detection
- Many later systems (including deep learning ideas) build on these concepts

Paper 3: Dimensionality Reduction for Handwritten Digit Recognition

AUTHORS: S. SUCHITRA ET AL.

1. What is the main goal of this paper?

The main goal of this paper is to reduce the number of features used in handwritten digit recognition without losing accuracy, and at the same time reduce computation time and memory usage.
This problem is known as the curse of dimensionality:

- Too many features → slow training
- Higher risk of overfitting
- Harder to interpret the model

The paper investigates how dimensionality reduction techniques can solve this problem effectively.

2. Why is dimensionality reduction important here?

Handwritten digits:

- Have many variations in writing style
- Require rich feature extraction (like HOG or Gabor)
- This leads to very high-dimensional feature vectors

High-dimensional data:

- Needs more memory
- Needs more computation
- Can hurt generalization

➤ Dimensionality reduction keeps important information and removes redundant or noisy features.

3. Overall system pipeline

The paper follows this pipeline:

1. Input datasets
   - MNIST (standard digits)
   - CVL Single Digit (more challenging)
2. Feature extraction
   - HOG (shape and edge-based)
   - Gabor filters (texture and frequency-based)
3. Dimensionality reduction
   - PCA (unsupervised)
   - LDA (supervised)
   - Isomap (non-linear)
4. Classification
   - SVM with RBF kernel
5. Evaluation
   - Accuracy
   - Processing time

Paper 3: Dimensionality Reduction for Handwritten Digit Recognition

Authors: S. Suchitra et al.

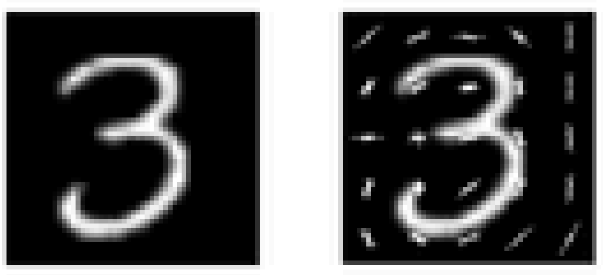
4. Feature extraction methods

HOG (Histogram of Oriented Gradients)

- Extracts edge direction information
- Focuses on shape
- Very effective for digits

Gabor filters

- Capture texture and stroke patterns
- Use multiple orientations and scales
- Produce very large feature vectors



5. Dimensionality reduction techniques explained

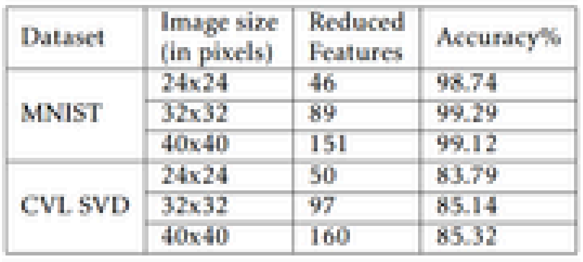
PCA (Principal Component Analysis)

- Unsupervised
- Keeps directions with maximum variance
- Reduces features while preserving most information

Result:

- Highest accuracy (99.29%)
- Moderate reduction

Table 2. Accuracy results for feature sets generated using HOG with PCA dimensionality reduction with classification using SVM with RBF kernel



Dataset	Image size (in pixels)	Reduced Features	Accuracy%
MNIST	24x24	46	98.74
	32x32	89	99.29
	40x40	151	99.12
CVL SVD	24x24	50	83.79
	32x32	97	85.14
	40x40	160	85.32

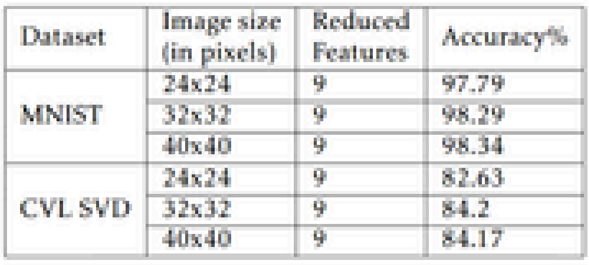
LDA (Linear Discriminant Analysis)

- Supervised (uses class labels)
- Maximizes class separability
- Produces at most (number of classes – 1) components
- → 10 digits → 9 components

Result:

- Slightly lower accuracy (98.34%)
- Extreme compression
- Very fast

Table 3. Accuracy results for feature sets generated using HOG with LDA dimensionality reduction with classification using SVM with RBF kernel



Dataset	Image size (in pixels)	Reduced Features	Accuracy%
MNIST	24x24	9	97.79
	32x32	9	98.29
	40x40	9	98.34
CVL SVD	24x24	9	82.63
	32x32	9	84.2
	40x40	9	84.17

6. Key experimental findings

Best accuracy

- HOG + PCA
- 99.29% accuracy on MNIST
- Comparable to CNN performance

Most efficient model

- HOG + LDA
- Only 9 features
- 75% faster
- Slight drop in accuracy (~1%)

Paper 3: Dimensionality Reduction for Handwritten Digit Recognition

AUTHORS: S. SUCHITRA ET AL.

7. Accuracy vs Efficiency trade-off

Model	Accuracy	Speed	Feature Count
HOG + PCA	Highest	Medium	~89
HOG + LDA	Very high	Fastest	9
CNN	High	Slow	Very large
Isomap	Good	Slow	Large

8. final conclusion

This paper shows that:

- Traditional ML + smart feature engineering
- Can compete with deep learning
- With far less computation

It proves that dimensionality reduction is not optional, but essential for real-time and embedded systems.

Final conclusion of the paper

- HOG + PCA → best accuracy
- HOG + LDA → best efficiency
- Dimensionality reduction dramatically improves performance
- Suitable for resource-constrained environments

Paper 4: Handwritten Digit Recognition System Using MCS HOG Features

AUTHORS: RESEARCH ON MULTIPLE-CELL SIZE HOG

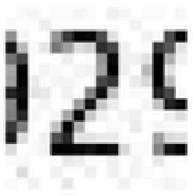
1. What is the main idea of this paper?

The paper proposes a high-accuracy handwritten digit recognition system that:

- Uses hand-crafted features (HOG) instead of deep learning
- Introduces a new feature extraction idea called Multiple-Cell Size HOG (MCS-HOG)
- Uses SVM for classification
- Achieves 99.36% accuracy on MNIST, which is comparable to CNNs

➤ The key contribution is improving HOG features, not inventing a new classifier.

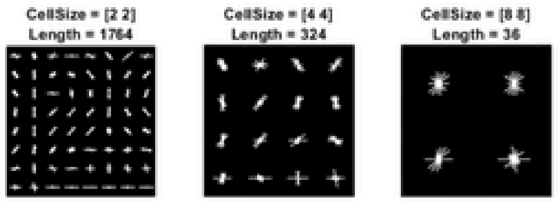
2. What problem are the authors solving?



Problem:

Handwritten digits vary a lot:

- Different writing styles
- Different stroke thickness
- Different shapes



Standard HOG:

- Is very sensitive to cell size
- A single cell size may:
  - Miss important details
  - Or produce too many confusing features

Goal:

Design a system that:

- Extracts stronger features - Keeps computation simple
- Avoids complex CNN architectures

3. The key contribution: MCS-HOG

Why normal HOG is not enough

HOG divides the image into small cells and computes gradient histograms. But:

- Small cells → too many features
- Large cells → lose detail

There is no single perfect cell size.

What is MCS-HOG?

Instead of choosing one cell size, the authors use two:

Cell Size	What it captures
6 × 6	Fine details
7 × 7	Slightly coarser structure

Then:

- Compute HOG features twice - Concatenate both feature vectors

➤ This creates a richer and more discriminative feature representation.
Different cell sizes capture different information — combining them improves recognition.

Paper 4: Handwritten Digit Recognition System Using MCS HOG Features

AUTHORS: RESEARCH ON MULTIPLE-CELL SIZE HOG

4. Why feature concatenation helps

By combining two HOG feature sets:

- The system captures:
    - Local details - Global shape patterns
- The classifier receives more informative features
- Leads to higher accuracy than either size alone

This is why: - Single HOG ≈ 99.1% - MCS-HOG = 99.36%

5. Dataset and evaluation

Dataset: MNIST

- 70,000 images - 28 × 28 grayscale digits

Evaluation methods: 1. Independent test set 2. 10-fold cross-validation

This ensures:

- High accuracy -Good generalization - No overfitting

5. Dataset and evaluation

Dataset: MNIST

- 70,000 images - 28 × 28 grayscale digits

Evaluation methods: 1. Independent test set 2. 10-fold cross-validation

This ensures:

- High accuracy -Good generalization - No overfitting

7. Results

MCS-HOG beats both single cell sizes

Confusion Matrix

- Very high accuracy for most digits
- Digit 8 is the hardest to classify

This paper proves that:

- Smart feature engineering still matters
- Deep learning is not always necessary
- Classical ML can be:
    - Accurate - Efficient -Practical

Especially useful for:

- Embedded systems - Low-resource devices - Real-time OCR applications

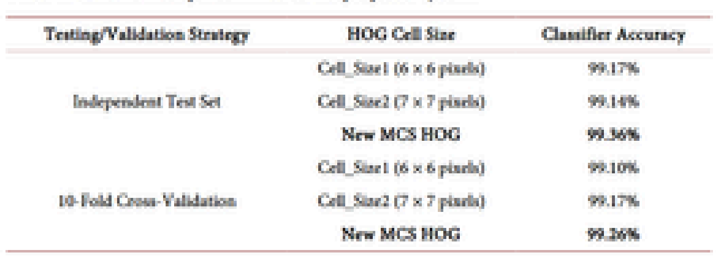


Table 2. Classification performance of the proposed system.

Testing/Validation Strategy	HOG Cell Size	Classifier Accuracy
Independent Test Set	Cell_Size1 (6 × 6 pixels)	99.17%
	Cell_Size2 (7 × 7 pixels)	99.14%
	New MCS HOG	**99.36%**
10-Fold Cross-Validation	Cell_Size1 (6 × 6 pixels)	99.10%
	Cell_Size2 (7 × 7 pixels)	99.17%
	New MCS HOG	**99.26%**

Paper 5: Using Random Forests for Handwritten Digit Recognition

Authors: Simon Bernard, Laurent Heutte, and Sébastien Adam (2007)

1. What is this paper about?

This paper studies Random Forests and answers a very practical question:
How should we choose Random Forest parameters to get good results for handwritten digit recognition?
Instead of proposing a new algorithm, the authors:

- Use an existing Random Forest method (Forest-RI)
- Test it on the MNIST handwritten digits dataset
- Carefully study how changing the parameters affects accuracy

So this paper is mainly about understanding behavior, not inventing something new.

2. What problem are they solving?

Handwritten digit recognition (digits 0–9) is difficult because:

- People write digits in many different styles
- Shapes vary a lot -Noise and distortions exist

Random Forests are known to work well, but:
❓ How many trees should we use?
❓ How many features should each tree look at when splitting?
Most people choose values like 100 trees without proof.
This paper tests these choices scientifically.

3. What algorithm do they use?

They use Random Forest – Forest-RI version, which works like this:
Random Forest idea

- Build many decision trees
- Each tree is trained on:
  - A random sample of training data (Bagging)
  - A random subset of features at each split (Random Subspace)
- Final prediction = majority vote of all trees

4. What are the two important parameters?

The whole paper focuses on two parameters only:
1. L = Number of trees

- How many decision trees are in the forest

2. K = Number of features per split

- At each node, the tree randomly selects K features and chooses the best one to split on

These two parameters control:

- Accuracy -Training time
- Diversity between trees

Paper 5: Using Random Forests for Handwritten Digit Recognition

AUTHORS: SIMON BERNARD, LAURENT HEUTTE, AND SÉBASTIEN ADAM (2007)

5. What data did they use?

They used MNIST, which contains:

- 60,000 training images - 10,000 test images
- Digits from 0 to 9 - Image size: 28 × 28 pixels

Instead of raw pixels, they extracted 84 simple grayscale features to focus only on Random Forest behavior.

6. What experiments did they run?

They tested many combinations of:

- L (trees) = 10, 50, 100, 150, 200, 300 - K (features) = 1 up to 84

For each (L, K):

- They trained 5 forests - Tested on the same test set
- Took the average accuracy

This makes the results reliable and fair.

7. What did they discover?

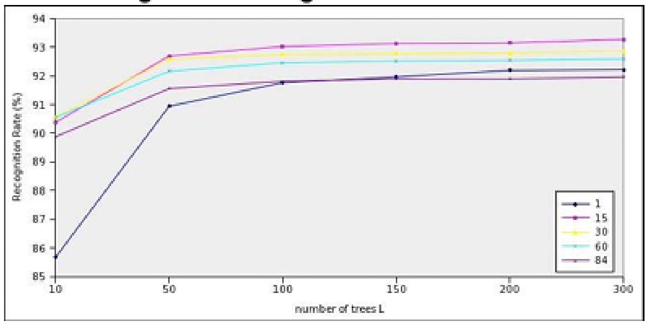
Effect of number of trees (L)

- Accuracy increases when L increases
- BUT the improvement is logarithmic:
  - Big improvement at first
  - Very small improvement after ~100 trees

Meaning:

Using more than 100 trees gives very little benefit but costs more computation.

Figure 2. Recognition rates wrt L



Effect of number of features (K)

The behavior of K is not linear:

Small K (K = 1)

- Trees are very random - Accuracy is poor
- Splits are almost random → weak trees

Medium K (K ≈ 5–20)

- Best performance
- Good balance between:
  - Tree strength - Tree diversity

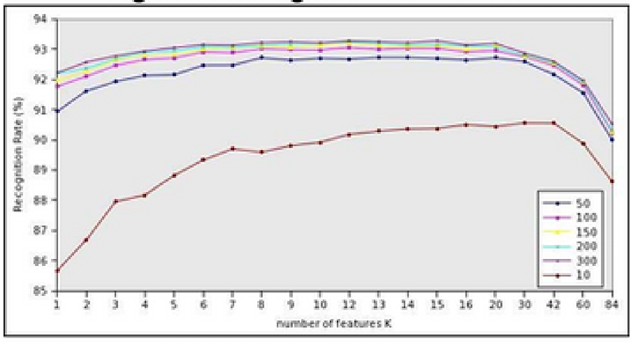
Large K (K > 20)

- Accuracy drops - Trees become too similar
- Less diversity → worse ensemble

Key idea:

Random Forests work well only if trees are different from each other.

Figure 3. Recognition rates wrt K



The paper identifies an optimal region, not a single value:

Best performance zone:

- L (trees): 100 – 300
- K (features): 5 – 20

Paper 6: Handwritten Character Recognition Using Neural Network Architectures

AUTHORS: AT&T BELL LABORATORIES (1990)

1. What is this paper about?

This paper is one of the early and important works on using neural networks for handwritten digit recognition, especially for real-world applications like reading US postal ZIP codes.

➤ The authors are not just asking:

"Can a neural network recognize digits?"

➤ They are asking:

"How can we make a neural-network system reliable, accurate, and practical for real usage?"

2. What problem are they trying to solve?

In real systems (like postal services):

- Wrong classification is expensive
- Rejecting uncertain digits is sometimes better than making a mistake

➤ So the system must balance:

- Accuracy (low error rate)
- Rejection rate (how many digits it refuses to classify)
- Cost (some mistakes are worse than others)

This paper focuses heavily on handling uncertainty intelligently.

3.Use of Neural Networks

- A feed-forward multi-layer neural network is used.
- The input is an image of a handwritten digit.
- The network learns to classify digits (0–9) using training examples.
- A single network already gives good accuracy, but it still makes mistakes.

4. Combining Multiple Neural Networks

- Instead of relying on one network, the paper combines several independently trained networks.
- Each network makes different mistakes.
- By averaging their outputs, the system becomes:
  - More accurate
  - Less likely to reject correct digits
- Even a weaker network can help when combined with others.

➤ Key idea: Multiple networks together perform better than any single one.

Paper 6: Handwritten Character Recognition Using Neural Network Architectures

AUTHORS: AT&T BELL LABORATORIES (1990)

5. Softmax for Probability Estimation

- The raw outputs of the network are converted into probabilities using Softmax.
- This tells how confident the system is about its decision.
- Higher Softmax score → higher chance the digit is correct.

➤ Key idea: Softmax makes the network's decision more reliable and meaningful.

6. Rejection Mechanism

- If the confidence is low, the system rejects the digit instead of guessing.
- This reduces serious mistakes.
- The rejection decision is based on the probability values from Softmax.

7. Cost-Aware Decision Making

- Not all mistakes are equally bad.
- For example:
- A wrong first digit in a ZIP code is very costly.
- A wrong last digit is less costly.
- The system considers this cost difference when deciding whether to accept or reject a digit.

➤ Key idea: Decisions should minimize real-world cost, not just error rate.

8. Final system structure

- Think of it as two stages:

Stage 1: Neural networks

- Recognize digits
- Output confidence scores

Stage 2: Decision logic

- Uses probabilities
- Applies rejection rules

This is better than using a neural network alone.