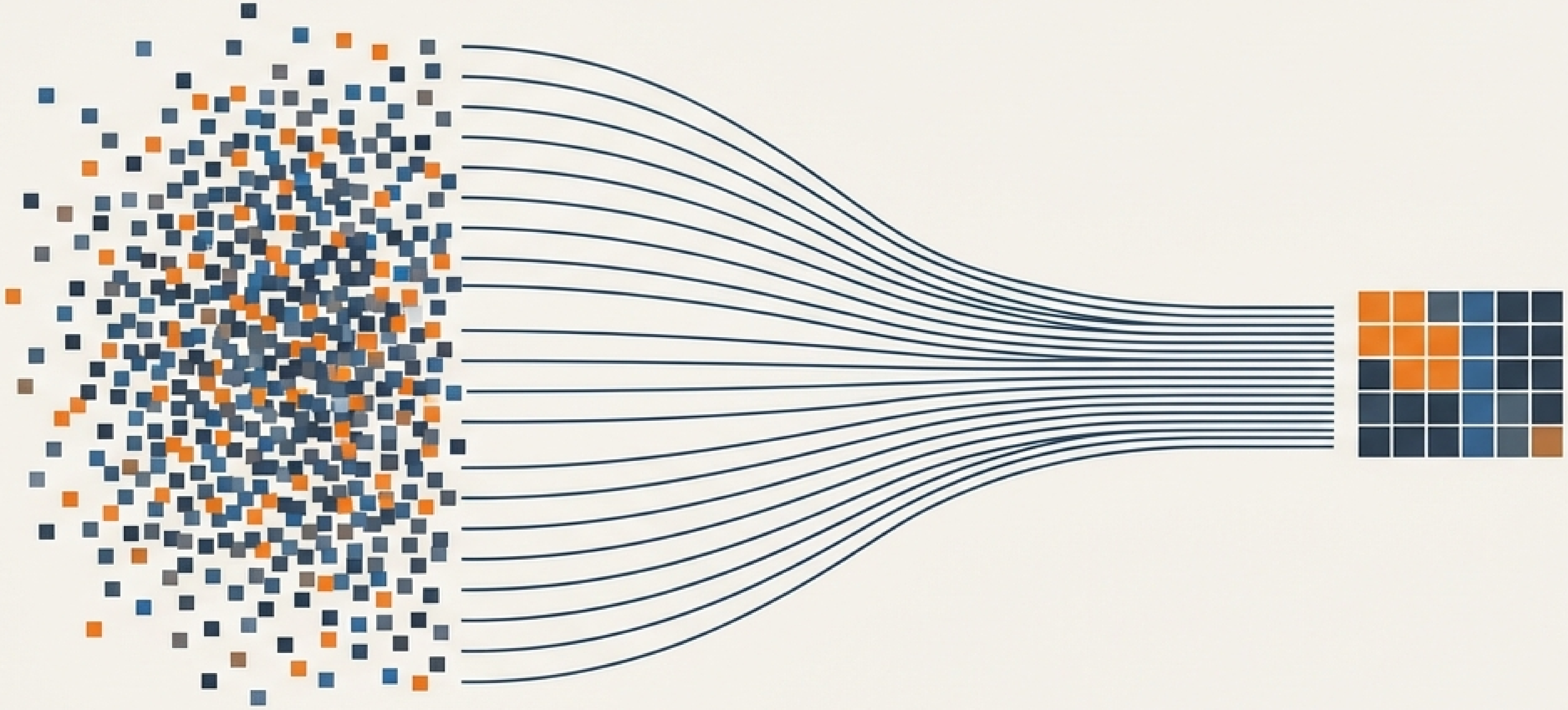


From Pixels to Insight

A Practical Guide to Machine Learning in OpenCV

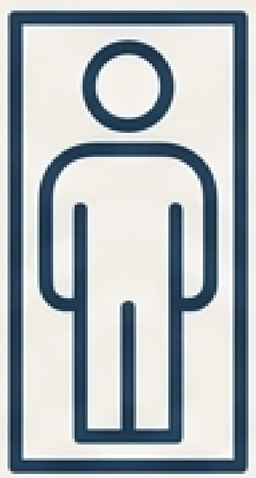
Our Goal: Turning Pixels into Action

The presentation is structured around three fundamental computer vision tasks.

1. Classification: *What is it?*

Assigning a single label to an entire image.

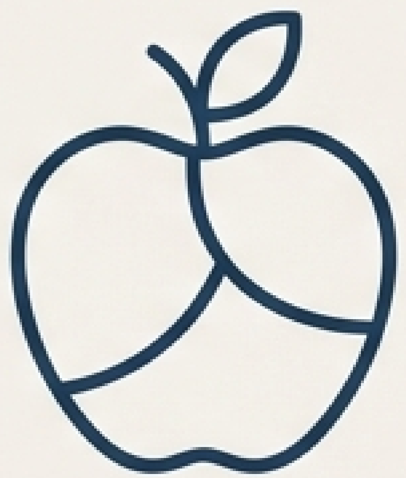
Example: Identifying a handwritten digit from the `digits.png` dataset.



2. Detection: *Where is it?*

Finding the location and shape of objects within an image.

Example: Locating pedestrians in a frame from the `vtest.avi` video.

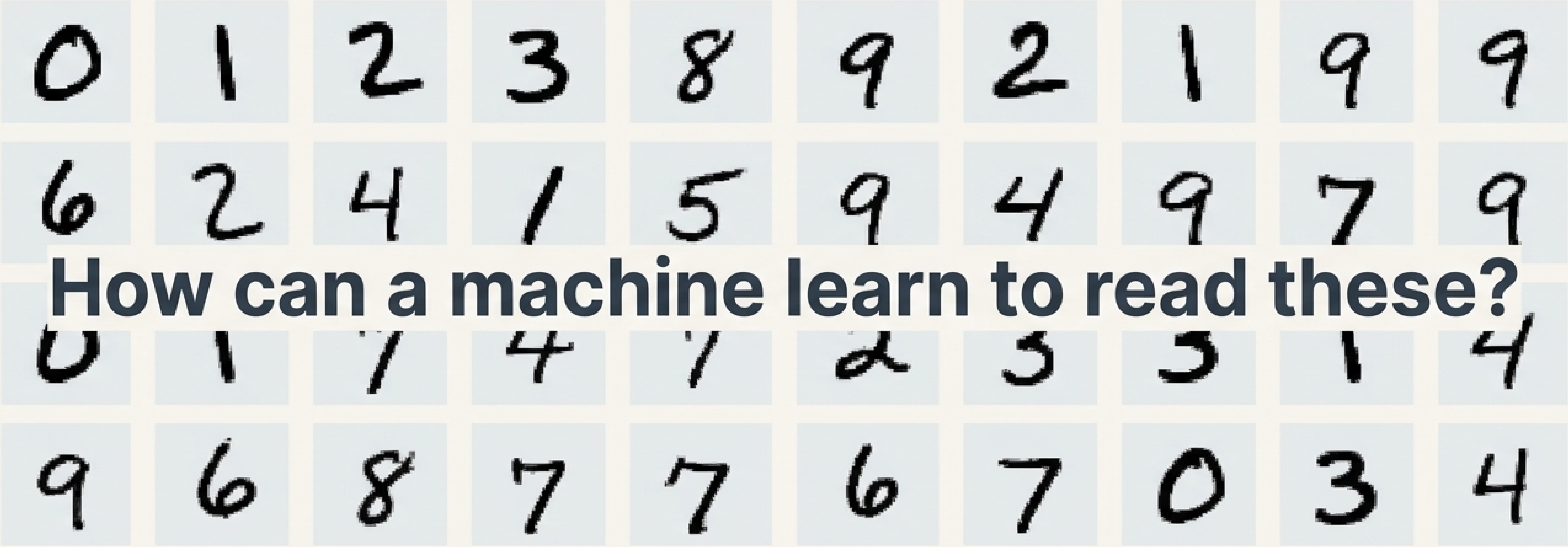


3. Segmentation: *What are its parts?*

Grouping pixels into meaningful regions or clusters.

Example: Isolating the dominant colors in the `fruits.jpg` image.

The Classification Challenge



How can a machine learn to read these?

Dataset: 5000 handwritten digits from `digits.png`.

Input: Each digit is a 20x20 pixel grid, flattened into 400 raw features.

Goal: Train a model that can correctly label new, unseen digits from a test set.

Three Approaches to Classification

**Democracy""

K-Nearest Neighbors (KNN)

Source Serif Pro

The label is decided by a majority vote from its closest known neighbors.

**Optimization"*

Support Vector Machine (SVM)

Source Serif Pro

The label is decided by finding the clearest possible boundary (hyperplane) that separates the classes.

**Interrogation""

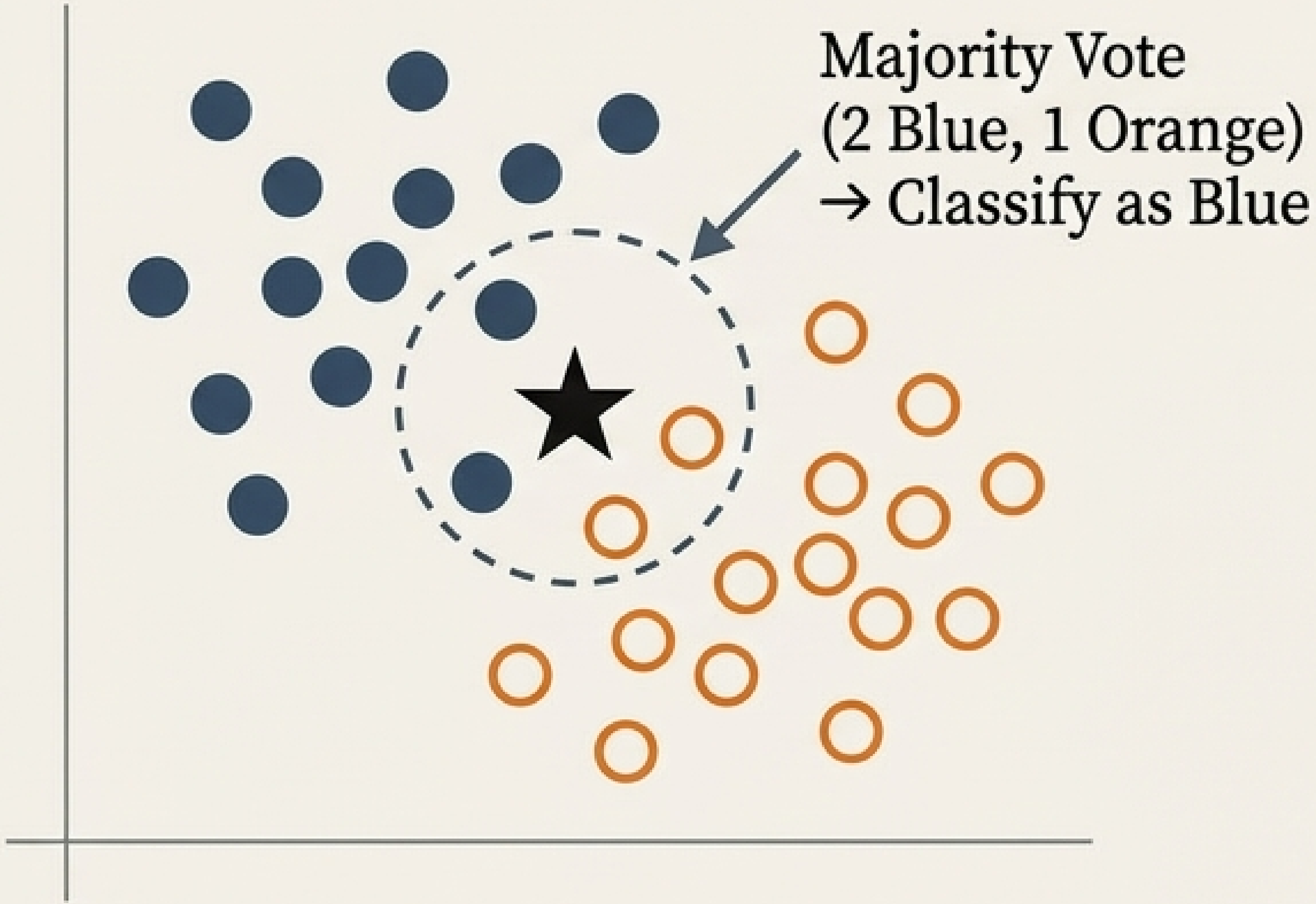
Decision Tree

Source Serif Pro

The label is decided by asking a series of simple "yes/no" questions about the pixel features.

Approach 1: K-Nearest Neighbors (KNN)

Classification by Proximity



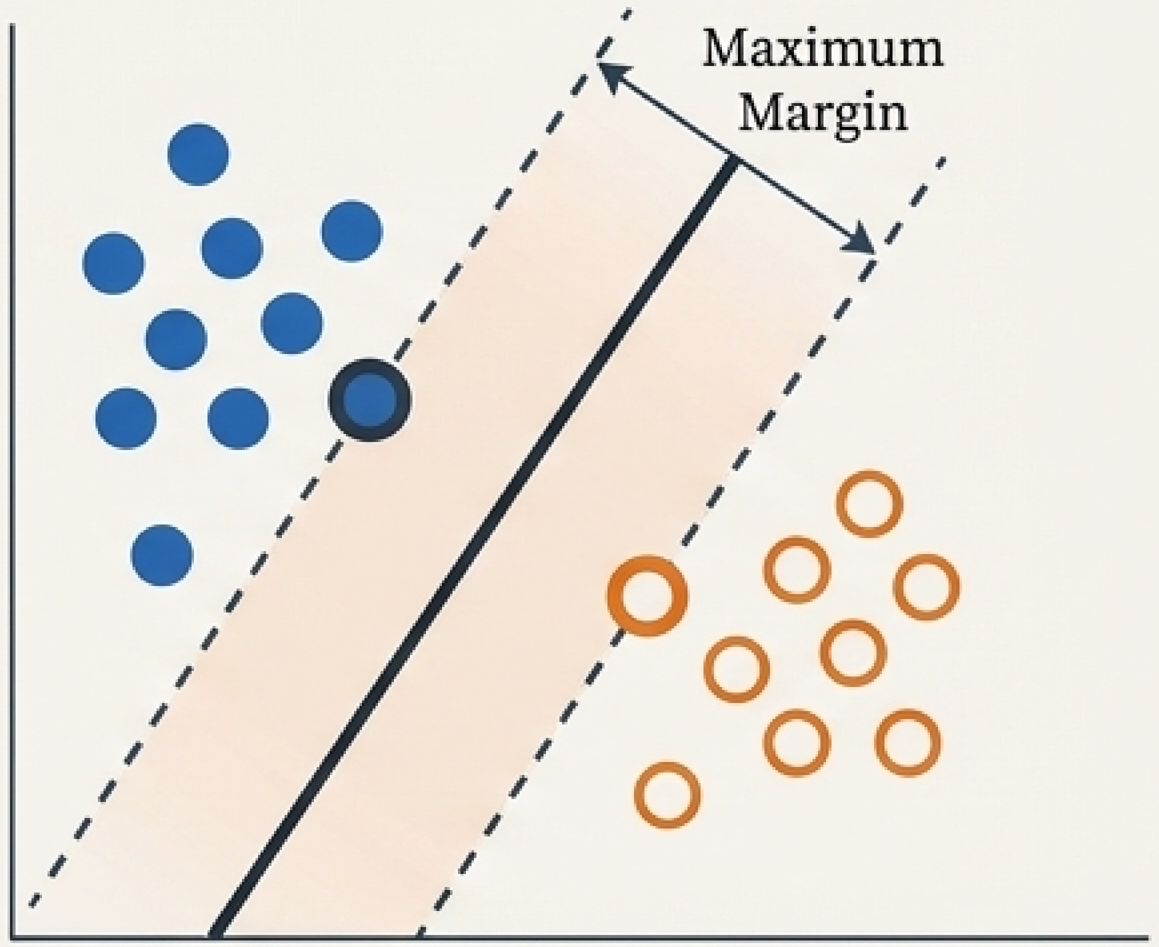
OpenCV Implementation

```
# Create, Train, and Predict
knn = cv2.ml.KNearest_create()
knn.train(train_data,
   cv2.ml.ROW_SAMPLE, labels)
ret, results, neighbors, dist =
   knn.findNearest(test_data, k=3)
```

Simple and intuitive, but prediction can be slow as it compares against all training data.

Approach 2: Support Vector Machine (SVM)

Finding the Maximum Margin



Uses Kernel Functions (like the default RBF kernel) to handle complex, non-linear boundaries.

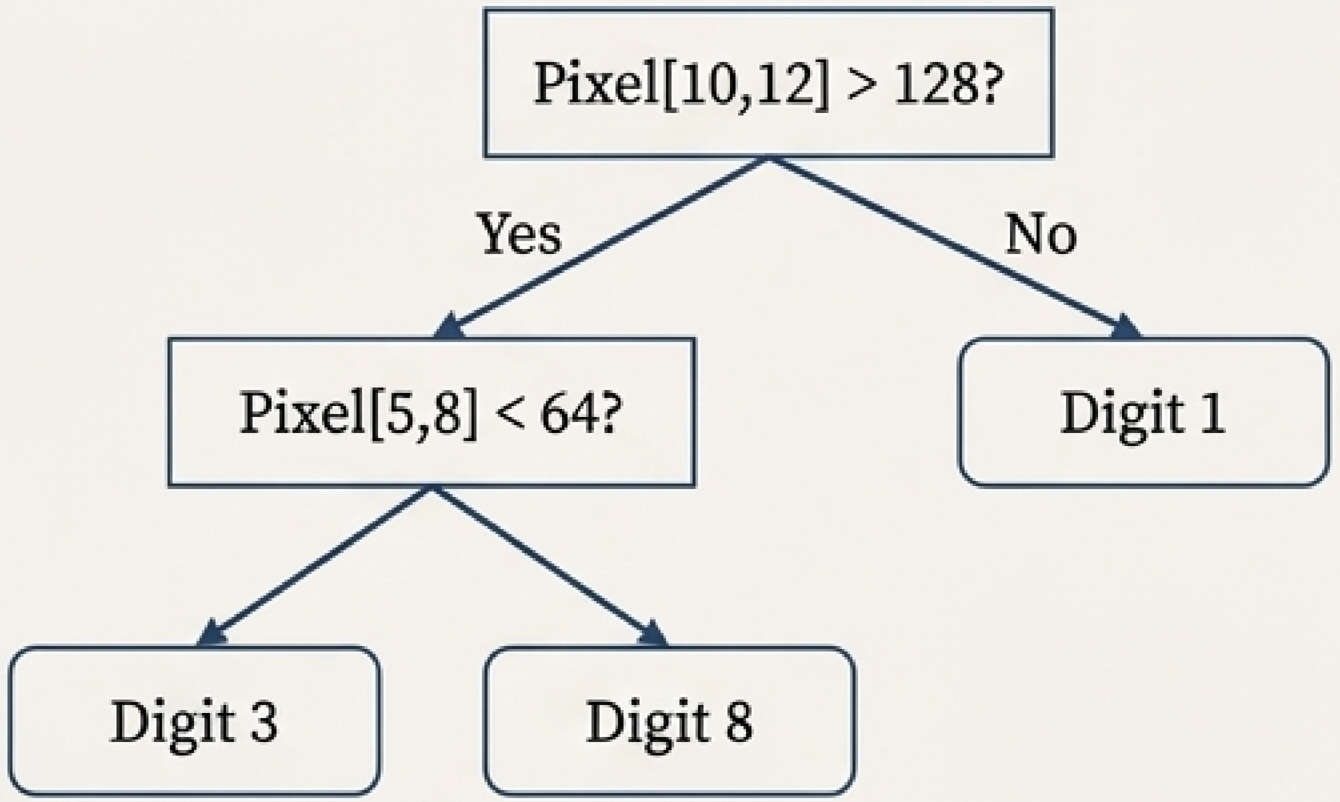
OpenCV Implementation

```
# Configure, Train, and Predict
svm = cv2.ml.SVM_create()
svm.setKernel(cv2.ml.SVM_RBF) # Powerful default
svm.setC(2.5)                 # Regularization
svm.train(train_data, cv2.ml.ROW_SAMPLE, labels)
_, prediction = svm.predict(test_data)
```

Powerful and accurate, especially for complex problems where classes are not easily separated.

Approach 3: Decision Tree

A Flowchart of Questions



Prone to overfitting. Control complexity with parameters like `MaxDepth` and `MinSampleCount`.

OpenCV Implementation

```
# Control Overfitting and Train
dtree = cv2.ml.DTrees_create()
dtree.setMaxDepth(10)  # Prevents overfitting
dtree.setMinSampleCount(5)
dtree.train(train_data, cv2.ml.ROW_SAMPLE, labels)
_, prediction = dtree.predict(test_data)
```

Fast and highly interpretable, but requires careful tuning to prevent overfitting to the training data.

The Verdict: Digit Recognition Performance

Algorithm	Accuracy	Prediction Speed	Interpretable
KNN	~96%	Slow	High
SVM (RBF)	**~98%**	**Fast**	Low
Decision Tree	~85%	Fast	**High**

For this task, **SVM provides the best balance of accuracy and speed.**

A New Challenge: From Pixels to People

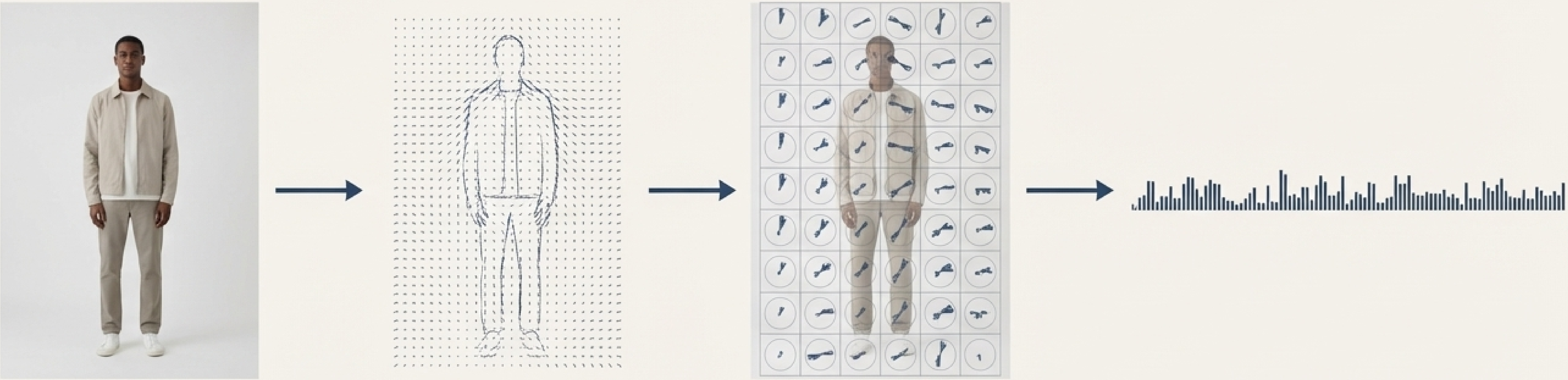
Classifying the whole image isn’t enough. How do we find objects *within* it?

The Problem: Raw pixel values are too variable to reliably describe a “person.” Factors like changing light, clothing color, and pose make simple classification fail.

***The Solution*:** We need a more robust representation of the object’s shape. We need to engineer *features*.

The Feature: Histogram of Oriented Gradients (HOG)

HOG creates a "fingerprint" of an object's shape by analyzing the direction of intensity changes (gradients). It is robust to lighting and small variations.



1. Input Image
2. Gradient Field
3. Cell Histograms
4. Final Feature Vector

In Action: Pedestrian Detection

HOG + SVM: A Powerful Combination

OpenCV provides a pre-trained HOG + SVM model specifically for detecting people.

```
# Load the default people detector and run it
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())
boxes, weights = hog.detectMultiScale(frame)
```

By combining a powerful feature descriptor (HOG) with a strong classifier (SVM), we can solve complex detection tasks.

The Final Challenge: Finding Structure Within

Instead of labeling the image, can we find its K dominant colors?

- **Task:** Group millions of pixels into a small number of clusters based on their color values. This is also known as 'color quantization.'
- **Approach:** Unsupervised Learning. We don't provide labels (like 'apple' or 'orange'); the algorithm discovers the most prominent color groups on its own.

The Tool: K-Means Clustering

Grouping by Similarity

fruits.jpg

quantized

OpenCV Implementation

```
# Reshape pixels and run K-Means
pixels = image.reshape(-1, 3).astype(np.float32)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
compactness, labels, centers = cv2.kmeans(
    pixels, K=8, None, criteria, 10, cv2.KMEANS_PP_CENTERS)
# Rebuild image from the K center colors
quantized_img = centers[labels.flatten()].reshape(image.shape)
```

K-Means is a versatile tool for discovering underlying groups in data, from colors to customer segments.

Your OpenCV ML Toolkit

CLASSIFY (What is it?)

Algorithms: KNN, SVM, Decision Trees

Best For: Problems with pre-labeled, categorized data (e.g., OCR, document classification).

DETECT (Where is it?)

Algorithms: HOG + SVM

Best For: Finding specific objects with defined shapes (e.g., pedestrians, cars).

SEGMENT (What are its parts?)

Algorithms: K-Means Clustering

Best For: Unsupervised grouping and data simplification (e.g., color quantization, medical image analysis).

Practical Essentials

- **Model Persistence:** Don't retrain every time. Save and load models using `model.save()` and `cv2.ml.load()`.
- **Validation:** Always measure performance on unseen test data. Cross-validation is your friend for robust evaluation.

The Tools Are Ready. What Will You Build?

Dive Deeper with Official OpenCV Tutorials

- OpenCV ML Documentation: https://docs.opencv.org/4.x/d6/de2/tutorial_py_table_of_contents_ml.html
- SVM In-Depth: https://docs.opencv.org/4.x/d1/d73/tutorial_introduction_to_svm.html
- K-Means for Segmentation: https://docs.opencv.org/4.x/d1/d5c/tutorial_py_kmeans_opencv.html
- HOG Descriptor Details: https://docs.opencv.org/4.x/d5/d33/structcv_1_1HOGDescriptor.html