# Pill Identification via Supervised Learning: A Hybrid CNN-SVM Model for Drug Recognition and Metadata Retrieval


1st Andrea Giraldo-Puerta
*Dept. of Mathematical Sciences*
*Stevens Institute of Technology*
Hoboken, United States
agiraldo1@stevens.edu

2nd David Arguello
*dept. of Computer Science*
*Stevens Institute of Technology*
Hoboken, United States
darguell@stevens.edu

3rd Juliana McGaffic
*dept. of Computer Science*
*Stevens Institute of Technology*
Hoboken, United States
jmcgaffi@stevens.edu



***Abstract*—Accurate pill identification is critical in both the pharmaceutical and medical sectors to ensure patient by healthcare professionals and non-professional caretakers. The objective of this study is to develop a machine learning algorithm based on the concepts learned in the curriculum to accurately identify pills based on images. CNN algorithms were applied (ResNet18, DenseNet121, EfficientNet-B0) for image classification of classes with more than 5 images with EfficientNet-B0 achieving the highest validation accuracy of 69.22% and F1 score of 65.09%. For few-shot classes, the algorithms implemented are SVM, k-NN, and PCA, with k-NN (k=1, 50 components) performing best at 17.69% accuracy. For condition mapping, Decision Trees were used on metadata and a TF-IDF + Logistic Regression model on enriched FDALabel text, with the latter achieving 98.9% accuracy and 90.5% Macro-F1. By combining classification via metadata with image classification, this solution demonstrates that hybrid modeling offers promising pathways for robust pill classification systems.**


## I. Introduction

In the pharmaceutical and healthcare industries, accurate medication identification is critical for patient safety, preventing clerical errors, and avoiding lawsuits. Aside from healthcare professionals, caretakers without medical training need to identify medications to deliver correct dosages. In accidental ingestion cases (by children, pets, or adults), proper pill identification significantly improves emergency outcomes.

This project leverages a pill image library and supervised learning to build an algorithm that identifies pills from user-submitted images, contextualizing them with associated conditions and symptoms.

Using the first database containing images of pills, a CNN model will be constructed to identify the type of pill. In case of classes with less than 5 images, an SVM model will be used as a fallback to the CNN model to improve the accuracy. From these two strategies, a pill class or name will be identified. The second dataset containing pill usage information will then be introduced to identify conditions which the pill is used to treat, the usage of the pill, and symptoms that an individual may be experiencing.

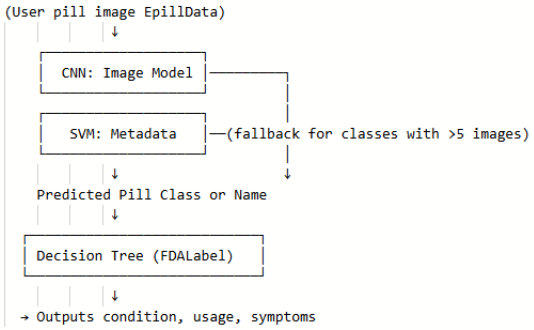


Fig. 1. Proposed Hybrid Pill Identification System Architecture

## II. Related Work

In 2020, Usuyama et al. proposed a solution, leveraging one of two NIH-provided datasets to address the challenges of practical pill identification. The dataset emphasizes the complexity of real-world conditions by incorporating consumer-grade images—photographs taken in varied lighting conditions, backgrounds, and using non-professional equipment—where many pills appear visually similar. The reported best performing model in this instance uses a multi-head metric learning approach. To address the "real-world" images, the model uses a CNN encoder followed by a bilinear transformation layer to enhance the fine-grained feature representation and effectively support low-shot classification tasks.

Ponte et al. proposed a solution in 2023 leveraging computer vision techniques to reduce medication dispensing errors. This approach uses a Keras-based deep learning model for pill image processing and uses ImageDataGenerator for image augmentation during preprocessing. To extract and analyze key pill characteristics such as shape, color, and imprint, the model uses OpenCV and Paddle OCR to improve the classification. One of the key features of this model, is that it was designed for real-time, high-volume use. The model interfaces with video cameras, making this a suitable solution specifically designed to target the pharmaceutical industry.

Lee et al. takes a similar approach but incorporates tools for object detection to locate pills in addition to identifying them based on a pill database. Like the solution proposed by Ponte et al., this system is built using Keras but also incorporating TensorFlow for image processing. OpenCV is used to facilitate the image preprocessing and Optical Character Recognition (OCR) systems assist in reading imprints.

In a 2024 solution proposed by Kavitha et al., the aim was to automate pill detection in both real-time and static images. The model uses deep learning techniques, specifically the YOLO-based object detection, MobileNet for feature extraction, and custom classification layers to predict pill types across 20 categories. When tackling the data preprocessing, the model uses rescaling, resizing, and data augmentation to enhance model generalization.

## III. Our Solution

### A. Description of Dataset

**Image Dataset – ePillID**
*Source:* CVPR 2020, Usuyama et al.
*Website:* ePill Website
*GitHub:* ePill Github

The ePillID dataset is a large-scale benchmark designed to evaluate fine-grained pill identification models under low-shot learning conditions. It addresses the real-world problem of matching consumer-quality pill photographs to standardized reference images of FDA-approved medications. Each pill is represented by two appearance classes (one for each physical side), and the dataset encourages learning under data-scarce scenarios—many classes have only one reference image.

**Size and Content:**

- Approximately 13,000 images
- 9,804 appearance classes representing 4,902 pills (each with two sides)
- Includes both reference images and real-world consumer photos
- Metadata: pill name, imprint text, shape, color, and National Drug Code (NDC)

**Use Case:** Designed to benchmark low-shot, fine-grained classification tasks with applications in pill recognition using metric learning and deep learning approaches.

**Data Preprocessing**

We applied several cleaning and preprocessing steps to ensure dataset integrity and prepare it for modeling:

- Column and String Cleaning:
  - Removed leading/trailing whitespace from column names and string values.
  - Ensured all image paths ended with .jpg and stripped extra spaces.
  - Dropped rows with missing critical fields (image_path, label).
- Path Construction and Validation:
  - Built a full_path column by joining the base directory path with image filenames.
  - Verified file existence on disk; removed entries with missing files.
- Label Encoding:
  - Encoded pill class labels into integer format (`label_encoded`) using `LabelEncoder`.
  - Saved the fitted encoder for decoding predictions during evaluation.
- Metadata Cleaning:
  - Trimmed whitespace from string/integer-based metadata (pilltype_id, label_code_id, prod_code_id).
  - Converted code IDs to numeric types where applicable.
  - Mapped Boolean fields (is_ref, is_front, is_new) from strings to binary integers (1 = True, 0 = False).
- Duplicate & Null Removal:
  - Identified and removed duplicate rows (none found).
  - Dropped any remaining rows with missing values in essential metadata fields.
- Dataset Split:
  - Performed an 80/20 train-validation split to support model development and evaluation.
- Final Cleaned Dataset Statistics:
  - Total usable images: 13,532
  - Unique classes: 4,902
  - Training set: 10,825 images
  - Validation set: 2,707 images
- Output Artifacts (Saved):
  - **cleaned_epillid_all_data.csv**: Cleaned image-level ePillID (.jpg only, valid paths, label + label_encoded)
  - **cleaned_epillid_metadata_only.csv**: Slim join set: pilltype_id, label_code_id, prod_code_id, label, label_encoded

**Data Exploration**

To provide quick insights into class balance and image quality/variety, we generated two lightweight visualizations from the cleaned dataset:

- Class Distribution (Top 10 Classes).
  - Computed the frequency of each pill class using the label_encoded column.
  - Sorted and plotted the top 10 most frequent classes to reveal dominant labels and potential imbalance.
  - This helps guide sampling strategies, class weighting, and evaluation choices.
- Sample Images Grid.
  - Drew a random sample of 10 images from the full_path column (with a fixed random_state for reproducibility).
  - Displayed them in a 2×5 grid to verify that paths are correct, images load, and visual characteristics (e.g.,

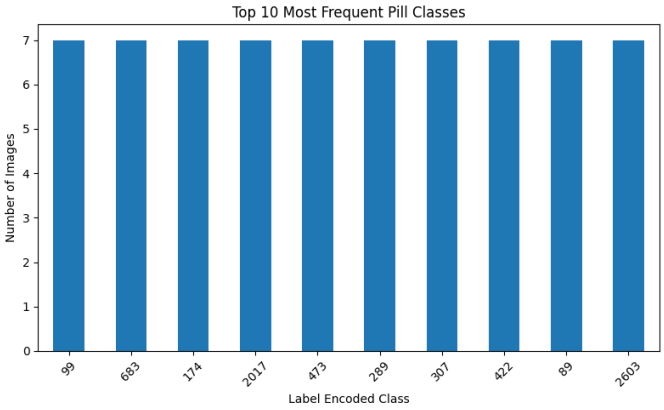


Fig. 2. Top 10 Most Frequent Pill Case (ePillID)

lighting, orientation, reference vs. consumer photos) match expectations.
- If any files were missing, a neutral placeholder image would render, flagging potential path issues (none occurred in the cleaned set).


Fig. 3. Sample Images (ePillID)

- Notes for Reproducibility.
  - The class plot uses value_counts().head(10) and a bar chart.
  - The image grid uses a fixed seed (random_state=42) to keep the same examples across runs.
  - These checks are non-destructive and operate on the already cleaned DataFrame (df) produced by the preprocessing pipeline above.

**Metadata Dataset – FDALabel**

*Source:* U.S. Food and Drug Administration (FDA)
*Website:* FDALabel Dataset (FDA)

FDALabel is a public resource offering structured access to the full-text labeling data submitted to the FDA for approved drugs. Each record contains rich details about the drug's intended uses, dosing guidelines, safety warnings, and pharmacological information.

**Size and Content:**

- Over 155,000 labeled drug entries
- Each record includes:
  - Brand and generic names
  - National Drug Code (NDC)
  - Indications and usage
  - Dosage and administration
  - Warnings and precautions
  - Adverse reactions
  - Drug interactions
  - Active/inactive ingredients
  - Manufacturer information

**Use Case:** Enables advanced querying and filtering of drugs by label sections, drug class, ingredients, or approval status. Supports downstream tasks such as mapping symptoms, classification of conditions, and interpretation of pill-related predictions.

**Data Preprocessing**

We cleaned and enriched the FDALabel dump to build a tidy, joinable table keyed by NDC elements and SPL Set IDs, then fetched optional label text from the openFDA API.

- Initial load and normalization
  - Loaded the raw CSV (fdalabel-query.csv) and stripped leading/trailing whitespace from all string columns.
  - Removed exact duplicate rows to reduce redundancy.
  - Starting size: 126,045 rows.
- Scope filtering (human/biologic only)
  - Retained HUMAN, VACCINE, and BIOLOGIC(S) labels; excluded ANIMAL/VETERINARY entries.
  - Post-filter rows 121,678. Rationale: align with human-use pill identification; exclude veterinary records.
- Canonical column names
  - Renamed common fields for consistency:
    * `Dosage Form(s)` → `dosage_form`
    * `Route(s) of Administration` → `route`
    * `Trade Name` → `brand_name`
    * `Generic/Proper Name(s)` → generic_name
    * `Company` → `company`
    * `NDC(s)` → `ndcs`
    * `FDALabel Link` → `fdalabel_link`
    * `DailyMed SPL Link` → `dailymed_link`
    * `SET ID` → `spl_set_id`
    * Key narrative sections: `Indications and Usage` → `indications`, `Warnings and Precautions` → `warnings`, `Adverse Reactions` → `adverse`
  - Lowercased key text fields (indications, warnings, adverse, brand_name, generic_name, route, dosage_form) for downstream NLP.
- Robust NDC extraction
  - Parsed 3-part NDCs (labeler-product-package, e.g., 12345-6789-00) and 2-part NDCs (labeler-product, e.g., 85160-100).

  - Avoided double counting by removing 2-part entries that are prefixes of detected 3-part NDCs in the same record.
  - Kept only rows with 1 detected NDC; 121,678 rows with at least one NDC.
  - Exploded to one NDC per row → increased rows when a record listed multiple NDCs.
- Key construction for joining
  - Split ndc into labeler_code, product_code, package_code (padding missing segments when only 2 parts).
  - Cast labeler_code and product_code to numeric (labeler_code_num, product_code_num) to serve as join keys with ePillID's code IDs.
  - Built a tidy table with: spl_set_id, brand_name, generic_name, company, route, dosage_form, ndc, labeler_code_num, product_code_num, indications, warnings, adverse, fdalabel_link, dailymed_link.
  - Deduplicated at the row level to avoid inflation from repeated links/text.
  - FDALabel cleaned table size: 148,084 rows (increase driven by NDC explosion).
- Label-text enrichment via openFDA
  - Sampled 5,000 unique spl_set_id values (out of 121,677 total) for API lookups (seeded for reproducibility).
  - Used requests_cache (SQLite, 7-day TTL) and exponential backoff for rate limits.
  - Collected and normalized free-text sections (indications, warnings, adverse): stripped, lowercased, whitespace-collated; truncated to 2,000 chars per field to cap extremes.
  - Dropped entries where all three text fields were empty; deduplicated by spl_set_id.
  - Left-joined enriched text back to the tidy table; missing text filled as empty strings.
  - Added has_text flag (True if any of the three sections is non-empty).
  - Fetched text rows: 5,000
  - Rows with any text: 3,581
  - Final enriched table size (after join): 148,084 (same cardinality as the tidy table; enrichment fills where available).
- Output artifacts (saved for downstream join with ePillID)
  - **fdalabel_clean.csv** (tidy, exploded by NDC, join keys ready)
  - **fdalabel_clean_enriched.csv** (same, plus openFDA text where available)
- Reproducibility quality checks
  - Deterministic sampling: random.seed(42).
  - API caching prevents redundant calls and stabilizes reruns.
  - Length truncation prevents outliers from skewing TF-IDF/vectorizers.
  - The has_text flag enables quick filtering for NLP experiments.
  - Numeric join keys (labeler_code_num, product_code_num) align with ePillID's code IDs for future merges.
- Final Cleaned/Enriched Statistics (FDALabel)
  - Rows after human/biologic filter: 121,678
  - Rows with 1 NDC: 121,678
  - Tidy exploded table: 148,084 rows
  - SPL set IDs sampled for text: 5,000 (of 121,677)
  - Enriched rows with any text: 3,581
  - Final enriched table size: 148,084

**Data Exploration**

To better understand the FDALabel dataset's composition and variability, we generated three exploratory visualizations:

- Top 10 Routes of Administration
  - Counted the frequency of unique route values across all enriched rows.
  - Plotted the top 10 most common administration routes as a bar chart to highlight dominant drug delivery methods.
  - This aids in identifying potential overrepresentation (e.g., oral or intravenous forms) that may influence downstream analysis.

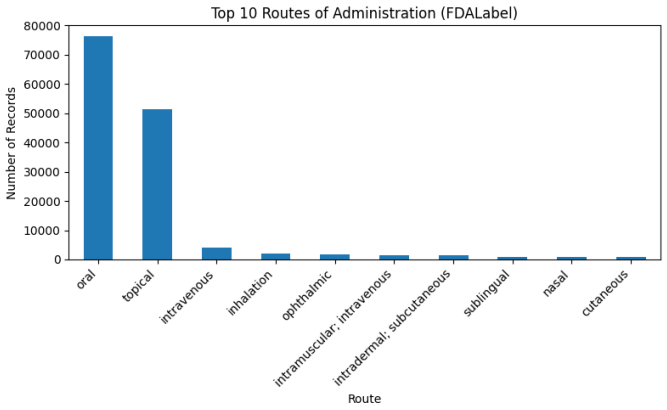


Fig. 4. Top 10 Routes of Administration (FDALabel)

- Top 10 Dosage Forms
  - Calculated the frequency of unique dosage_form values.
  - Visualized the top 10 dosage forms to understand formulation diversity and detect bias toward specific types (e.g., tablets, capsules, injectables).
- Histogram of NDCs per SPL Set ID
  - Before exploding NDCs into individual rows, grouped records by spl_set_id and counted the number of unique NDCs per SPL.
  - Plotted a histogram showing the distribution of NDC counts per SPL to reveal how often multiple product codes map to the same label set.

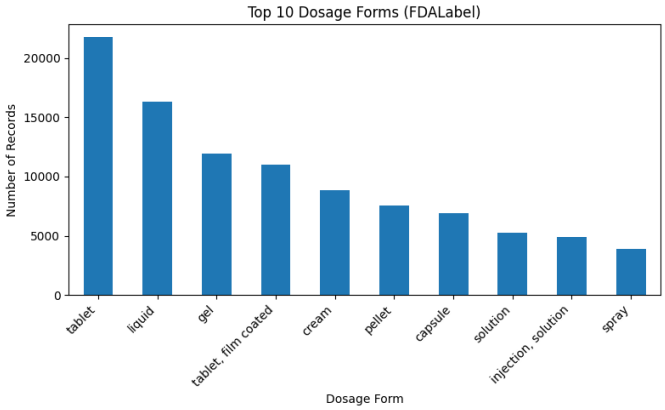


Fig. 5. Enter Caption

– This informs merging logic and highlights labeling complexity in certain drugs.

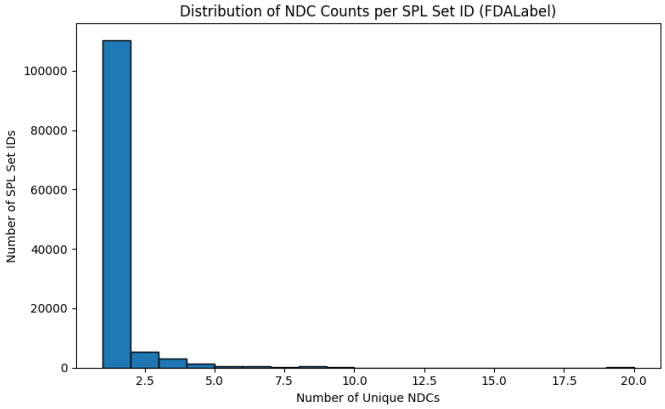


Fig. 6. Enter Caption

- Notes for Reproducibility
  - All plots are generated using the cleaned and enriched df_fda_enriched (or df_fda for pre-explode metrics).
  - Labels are rotated and chart layouts are tightened for clarity.
  - These checks serve both data quality validation (ensuring expected categorical values) and analysis readiness (understanding label structure before joining with ePillID).

### *B. Machine Learning Algorithms*

Our intelligent pill identification system leverages a hybrid architecture that integrates Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs), and Decision Trees (DTs)—each chosen for their strengths in handling different types of input and supporting robust prediction across varied data quality and sources.

#### 1. Convolutional Neural Networks (CNN)

The primary model for pill image classification is a Convolutional neural network, specifically a pretrained ResNet18 architecture. CNNs are particularly well-suited for visual recognition tasks due to their ability to extract hierarchical features from images. In this project, CNN identifies pills based on shape, color, and imprint from real-world and reference images in the ePillD dataset, which includes over 13,000 images across 4,902 pill classes.

**Why CNN?**

CNN are the standard for image classification tasks because they:

- Capture spatial hierarchies in visual data
- Are translation-invariant and robust to noise like blur or rotation
- Benefit from transfer learning: using pretrained weights (e.g., ResNet18 trained on ImageNet) allows effective learning even with relatively small medical images datasets.

In practice, the ResNet18 model was fine-tuned by replacing the final layer to match the number of pill classes. Data augmentation and dropout were used to prevent overfitting. The best model achieved over 93.07% training accuracy, though validation accuracy plateaued around 50.30%, indicating a need for further tuning.

#### 2. Support Vector Machines (SVM)

For pill classes that have fewer than 5 training images, we introduce a Support Vector Machine (SVM) as a fallback classifier. These rare pill types are excluded from CNN training but are still important to capture in a real-world system. Since CNNs tend to overfit classes with limited examples, the SVM offers a more generalized and stable solution for these low-sample categories.

We train the SVM using structured metadata—such as pill color, shape, and imprint text—which are readily available even when image quality is low. This makes the SVM a critical component in ensuring classification robustness across image quality levels.

**Why SVM?**

SVMs are effective for high-dimensional, small-to-medium-sized datasets and work well with structured data:

- They find optimal hyperplanes for classification.
- SVMs generalize well, even with limited data or in cases of class imbalance.
- They are especially helpful in fallback scenarios where image features are unreliable.

This dual-path strategy ensures robustness under real-world conditions like poor lighting, partial occlusion, or blurry images. According to Analytics Vidhya, SVMs can classify images effectively when combined with feature extraction techniques like Histogram of Oriented Gradients (HOG), which could be a future direction for our metadata pipeline.

The k-NN model is a supervised machine learning algorithm that is used for classification and regression problems. Predictions are made using a set number of closest training examples within the feature space. When the number of closest training samples set, the distance between the test and training samples are calculated, and, in the case of classification, the most frequent class among neighbors is selected. This strategy is useful for this problem due to its simplicity and flexibility. The dataset is very large and the algorithm needs to be adapted to handle small amounts of training data in different classes.

### 3. Decision Trees

Once a pill class or name is predicted by either the CNN or SVM, a Decision Tree classifier is used to map the pill to its associated usage information, conditions treated, and symptoms, using structured textual data from the FDA Label database. This data set provides structured, textual medical information aligned with each pill's National Drug Code (NDC).

**Why Decision Tree?**

Decision Trees are interpretable and ideal for structured datasets like the FDALabel:

- They allow clear traceability between a pill's classification and its medical purpose.
- The tree structure helps break down drug label information based on key features such as NDC, brand/generic name, and condition treated.
- They offer fast inference and are easy to update as more label data becomes available.

**Confidence-Based Filtering Strategy**

To enhance the precision of outputs from the Decision Tree, especially in ambiguous cases, we plan to implement a confidence-level thresholding approach. Specifically, we propose:

- Selecting the top 10 predictions whose probability values fall within the top 80% confidence range.
- This helps filter out low-confidence outputs and present only the most statistically reliable suggestions to end users (e.g., most likely medical conditions or usage scenarios for a given pill).

This approach is informed by the probability-based prediction capabilities of Decision Trees in scikit-learn, which allow access to .predictproba() for generating class likelihoods. For example, if a pill has multiple associated conditions, we can use the probability vector to return only those whose values exceed a dynamic threshold, such as the 80th percentile of confidence scores.

This confidence-filtered output ensures:

- Relevance: Only the most probable use cases are shown.
- Trust: Users receive more dependable insights.
- Scalability: As more label data is added, the tree can adapt without needing a complete retraining pipeline.

Resources like Fiveable's guide on decision trees and the scikit-learn DecisionTreeClassifier documentation support these applications, showing how tree depth, splitting criteria, and class probabilities can be tuned for optimal performance.

### *C. Implementation Details*

**CNN Implementation**

For the convolutional neural network (CNN) used for pill identification based on a pretrained ResNet, DenseNet, and EfficientNet architecture using PyTorch. The models were adapted for pill identification by replacing the final connected layer to match the number of unique pill classes in the ePill database. This approach uses transfer learning, which allows faster convergence and improved performance on smaller datasets. We also removed underrepresented classes with fewer than 5 examples to preserve class balance.

```
Classes with at least:
≥ 20 samples: 0
≥ 10 samples: 0
≥ 5 samples:  789
≥ 2 samples:  4902
```

Fig. 7. Shows the number of classes and their sample size.

**Training**

The CNN architecture consists of stacked convolutional layers followed by ReLU activations, max pooling, and dropout layers to mitigate overfitting. Fully connected layers followed by a SoftMax activation at the output are used for multi-class classification. The model was trained using the Adam optimizer with an initial learning rate of 0.001, batch size of 64, and cross-entropy loss.

**Hyperparameter Tuning**

We started by using a learning rate of 0.001, for which we had to take into consideration that higher values will lead to unstable training and lower values would lead to slowed convergence.

The batch size was chosen between 32 and 64, which 64 provided a better convergence speed. A dropout rate of 0.5 was used to reduce overfitting. Finally, 25 epochs were conducted in training due to underfitting with 10 and overfitting with anything higher than 25.

**Performance Evaluation**

Model performance was tracked using training and validation accuracy, loss, F1 score, recall, and precision. Both training and validation metrics improved gradually, indicating

the model was learning. For ResNet18, the highest achieved accuracy was 91.91% training and 43.43% validation at epoch 22, after which minor declines suggested the onset of overfitting.

In addition to ResNet18, we trained DenseNet121 and EfficientNet-B0 under the same settings.

- ResNet18 showed rapid convergence in training but plateaued on validation metrics around 43.43% accuracy, suggesting overfitting.
- DenseNet121 converged more slowly and achieved lower validation accuracy of 39.44%, likely due to its deeper architecture requiring more data or stronger regularization.
- EfficientNet-B0 consistently outperformed the other two architectures, reaching a peak validation accuracy of 69.22% and F1 score of 65.09%, demonstrating strong generalization capability. Its efficient parameterization allowed better feature extraction without overfitting, making it the most promising model for this task as shown in figure 9 and 10.

```
Epoch 1/25 | Train Loss: 7.5726, Acc: 0.0015 | Val Loss: 6.0992, Acc: 0.0120
Epoch 2/25 | Train Loss: 5.6582, Acc: 0.0169 | Val Loss: 5.4077, Acc: 0.0259
Epoch 3/25 | Train Loss: 4.7002, Acc: 0.0448 | Val Loss: 4.4879, Acc: 0.0717
Epoch 4/25 | Train Loss: 4.1025, Acc: 0.0877 | Val Loss: 4.1033, Acc: 0.1076
Epoch 5/25 | Train Loss: 3.6288, Acc: 0.1313 | Val Loss: 3.8382, Acc: 0.1295
Epoch 6/25 | Train Loss: 3.2567, Acc: 0.1841 | Val Loss: 3.8940, Acc: 0.1604
Epoch 7/25 | Train Loss: 2.8726, Acc: 0.2596 | Val Loss: 3.5791, Acc: 0.2032
Epoch 8/25 | Train Loss: 2.5500, Acc: 0.3142 | Val Loss: 3.1983, Acc: 0.2380
Epoch 9/25 | Train Loss: 2.1694, Acc: 0.3979 | Val Loss: 3.0790, Acc: 0.2809
Epoch 10/25 | Train Loss: 1.8140, Acc: 0.4863 | Val Loss: 2.9100, Acc: 0.3187
Epoch 11/25 | Train Loss: 1.5728, Acc: 0.5468 | Val Loss: 3.1434, Acc: 0.3088
Epoch 12/25 | Train Loss: 1.3561, Acc: 0.5974 | Val Loss: 2.8896, Acc: 0.3476
Epoch 13/25 | Train Loss: 1.0736, Acc: 0.6846 | Val Loss: 2.8275, Acc: 0.3606
Epoch 14/25 | Train Loss: 0.9031, Acc: 0.7297 | Val Loss: 2.6548, Acc: 0.4074
Epoch 15/25 | Train Loss: 0.8082, Acc: 0.7668 | Val Loss: 2.6359, Acc: 0.4054
Epoch 16/25 | Train Loss: 0.6549, Acc: 0.8114 | Val Loss: 2.6626, Acc: 0.4173
Epoch 17/25 | Train Loss: 0.5625, Acc: 0.8361 | Val Loss: 2.3777, Acc: 0.4442
Epoch 18/25 | Train Loss: 0.4692, Acc: 0.8622 | Val Loss: 2.4851, Acc: 0.4592
Epoch 19/25 | Train Loss: 0.4073, Acc: 0.8762 | Val Loss: 2.6417, Acc: 0.4532
Epoch 20/25 | Train Loss: 0.3760, Acc: 0.8876 | Val Loss: 2.5972, Acc: 0.4482
Epoch 21/25 | Train Loss: 0.3297, Acc: 0.9051 | Val Loss: 2.4426, Acc: 0.4681
Epoch 22/25 | Train Loss: 0.2823, Acc: 0.9240 | Val Loss: 2.4524, Acc: 0.4721
Epoch 23/25 | Train Loss: 0.2453, Acc: 0.9307 | Val Loss: 2.3481, Acc: 0.5030
Epoch 24/25 | Train Loss: 0.2149, Acc: 0.9400 | Val Loss: 2.7212, Acc: 0.4592
Epoch 25/25 | Train Loss: 0.1998, Acc: 0.9439 | Val Loss: 2.4359, Acc: 0.4791
```

Fig. 8. Sample of Training and validation metrics.

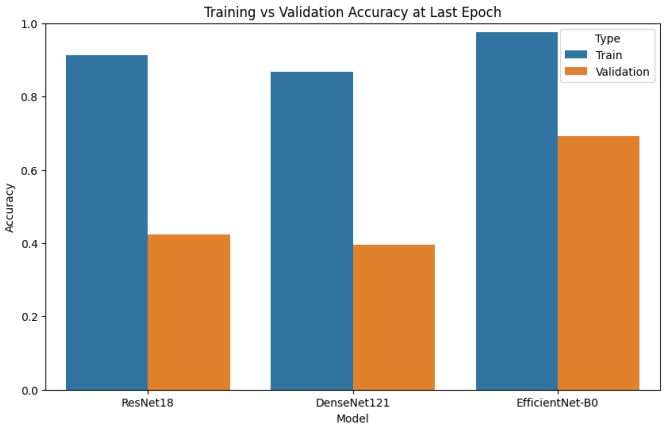


Fig. 9. CNN models comparison

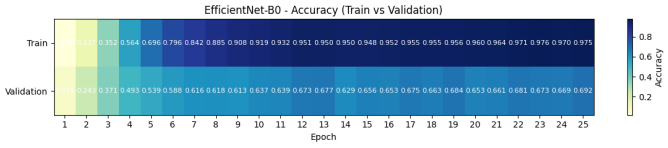


Fig. 10. EfficientNet-B0 Heat map Training Vs Validation accuracy

These results emphasize the importance of choosing pretrained architectures that balance model complexity and generalization for smaller, imbalanced datasets.

### Techniques and improvement

To improve the performance and prevent overfitting in the model, the techniques below were used:

- Data augmentation: applied to the training images using random horizontal flipping and rotation. This increases the diversity of the samples to help the model generalize unseen images.
- Transfer leaning: use of pretrained models, placing only the final classification layer, allowing the model to benefit from representations from ImageNet.
- Stratifies class filtering and balancing: Made sure the models did not bias towards underrepresented classes. As mentioned before, classes with fewer than five images were excluded to stabilize training.
- Validation monitoring: Used to track accuracy and loss trends for each epoch to help identify potential underfitting or overfitting.

The results indicate meaningful learning, but they need further improvement that will be discussed later in this paper.

### SVM Implementation

This section of the pill image classifier implementation addresses the handling of image classes with less than 5 images since those with 5 or more images are handled using the CNN algorithm. In all classes with fewer than 5 images, there are only 2 images per class, making the development of a machine learning algorithm particularly challenging. The size of the dataset inherently presents potential issues related to overfitting and low accuracy. Regarding testing the trained model, splitting the dataset into training and testing data also poses risks since there are only 2 images per class. Additionally, this model has a large amount of data to be trained, with 86% of classes containing less than 5 images. This aspect of the problem introduces the issue of runtime, with certain models and their hyperparameters struggling to converge due to the amount of data.

The first algorithm that was tested for this set of data is the SVM model. In the first iteration of testing using this model, the SVC in skit-learn was tested using a linear kernel. This model optimizes via quadratic programming and uses a one-vs-one strategy for multiclass classification. Immediately

when testing this model against the data, the model could not converge in a reasonable timeframe. To try to reduce the runtime using this model, the image resolution was significantly reduced to 8x8 pixels. This allowed the model to converge withing a reasonable amount of time but sacrificed the accuracy of the model. Another strategy applied to the SVC model to try to increase the resolution of the image while still getting the model to converge was applying principal component analysis (PCA) to reduce the dimensionality of the data being processed. Despite, applying PCA of varying factors to the input data, only reducing the resolution of the images allowed the model to converge with a low accuracy.

Using the learnings from applying SVC to the input data, the approach to this dataset shifted to using a model that is specifically able to effectively process large datasets.

Instead of using SCV, skit-learn's LinearSVC was applied to the dataset. Like the previous attempt at modeling this data, LinearSVC performs a linear classification using support vector machines, however, uses a one-vs-rest multiclass strategy and optimizes using coordinate descent. This makes LinearSVC better equipped to handle large datasets.

Although the LinearSVC is inherently better at handling large data sets, PCA was still applied to the input images to reduce the dimensionality and improve runtime. This allows the image resolution to be increased up to 64x64 pixels and ultimately increase the model accuracy. The number of components for PCA were tested for the runtime and accuracy of the classification using cross-validation. Additionally, the F-1 score, precision, and recall were calculated for each new value of components. The highest accuracy was calculated to be 17.69% using 60 components, however, the optimal number of components is 50 since the accuracy from 50 to 60 components only increases by 0.34% but the runtime increases by 28.5%.

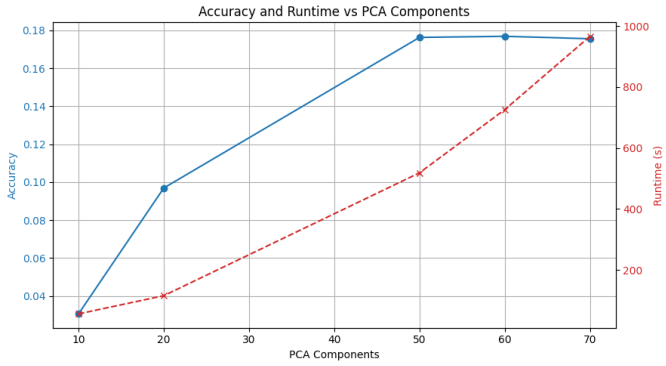


Fig. 11. Graph comparing the Accuracies and Runtimes using different numbers of PCA components.

With the best accuracy achieved so far being around 18% using SVM to conduct few-shot classification on the remaining data, other strategies were explored for handling these classes. The next strategy for handling classes with 2 images that was tested was the k-NN algorithm while maintaining the same image resolution. Using Euclidean distance and a k number of nearest neighbors, the optimal hyperparameters were tested using different number of PCA components and the number of nearest neighbors to consider, k. The results are shown in the table and heat map below.

Summary of Metrics by PCA Dimension:

| | PCA Components | Accuracy | F1 Score | Precision | Recall | Runtime (s) |
|---|---|---|---|---|---|---|
| 0 | 10 | 0.0305 | 0.0109 | 0.0079 | 0.0289 | 56.2755 |
| 1 | 20 | 0.0969 | 0.0584 | 0.0479 | 0.0971 | 115.9185 |
| 2 | 50 | 0.1763 | 0.1422 | 0.1290 | 0.1777 | 518.7239 |
| 3 | 60 | 0.1769 | 0.1451 | 0.1325 | 0.1784 | 725.8493 |
| 4 | 70 | 0.1756 | 0.1462 | 0.1346 | 0.1768 | 966.2109 |

Fig. 12. Table of results for the Accuracy, F-1 score, Precision, Recall and Runtime using different PCA components.

```
PCA Components: 50
Testing k = 1... Accuracy: 0.2982 | F1: 0.2738 | Time: 0.70s
Testing k = 3... Accuracy: 0.1488 | F1: 0.0949 | Time: 0.64s
Testing k = 5... Accuracy: 0.0956 | F1: 0.0465 | Time: 0.64s
Testing k = 7... Accuracy: 0.0707 | F1: 0.0287 | Time: 0.64s

PCA Components: 100
Testing k = 1... Accuracy: 0.2901 | F1: 0.2645 | Time: 0.74s
Testing k = 3... Accuracy: 0.1492 | F1: 0.0948 | Time: 0.70s
Testing k = 5... Accuracy: 0.0946 | F1: 0.0459 | Time: 0.72s
Testing k = 7... Accuracy: 0.0711 | F1: 0.0288 | Time: 0.75s

PCA Components: 150
Testing k = 1... Accuracy: 0.2831 | F1: 0.2556 | Time: 0.88s
Testing k = 3... Accuracy: 0.1482 | F1: 0.0941 | Time: 0.78s
Testing k = 5... Accuracy: 0.0949 | F1: 0.0464 | Time: 0.78s
Testing k = 7... Accuracy: 0.0718 | F1: 0.0288 | Time: 0.81s

PCA Components: 200
Testing k = 1... Accuracy: 0.2805 | F1: 0.2521 | Time: 0.93s
Testing k = 3... Accuracy: 0.1472 | F1: 0.0922 | Time: 1.25s
Testing k = 5... Accuracy: 0.0940 | F1: 0.0458 | Time: 1.30s
Testing k = 7... Accuracy: 0.0719 | F1: 0.0289 | Time: 0.88s
```

Fig. 13. Output of k-NN model testing using different values for PCA components and k.

The highest reported accuracy for the k-NN model occurred with 50 components and k=1. This is unsurprising that the highest accuracy occurs at k=1 given the dataset having only 2 images per class. When weighing the impact of higher values of k, the variance is significantly reduced, however, the accuracy suffers. With so little data per class, the model is likely overfitted as evidence my the cross-validation. Since accuracy is collectively being used as the primary evaluation metric for the success of the model, so far the most optimized model is the k-NN with k=1 and using PCA with 50 components.

### Decision Tree Implementation

We cast pill condition prediction as a nine-class, multi-class classification problem spanning "allergy", "cns", "cv", "derm", "endocrine", "gi", "infection", "pain", and "other". Two complementary model families were trained.

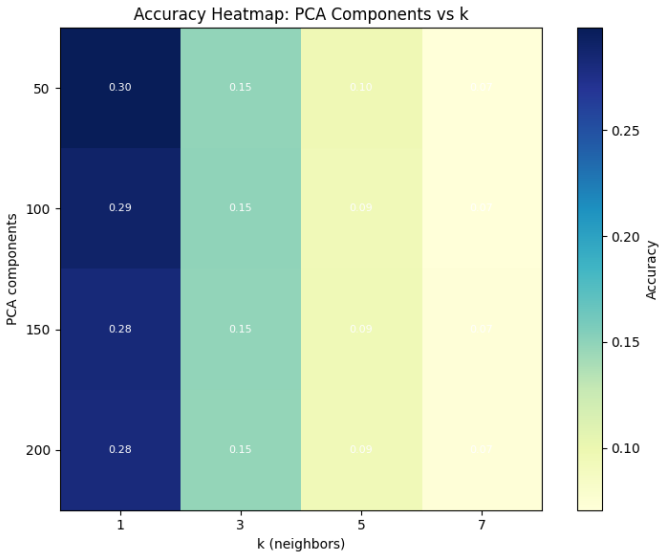


Fig. 14. Heat map showing results of k-NN model testing using different values for PCA components and k.

- **Text Model:** Utilized a TF-IDF vectorizer with 1–2 grams (`min_df`=5, `max_features`=5000), followed by Logistic Regression with balanced class weights.
- **Tabular Model:** Employed a Decision Tree trained on engineered metadata features, including one-hot encodings of `route` and `dosage_form` from FDALabel, as well as a compact set of keyword indicators extracted from the label text.

To stabilize evaluation, we used stratified train/validation splits and reported both Accuracy and Macro-F1.

### Join coverage and data linkage

Early experiments were hampered by the limited linkage between `ePillID` and `FDALabel`: only about 39.5% of `ePillID` rows matched `FDALabel` via numeric keys, which deprived the Decision Tree of metadata. We addressed this by collapsing `FDALabel` to a single row per `labeler_code_num` and `product_code_num`, keeping the record with the richest text, and then adding a fuzzy backup join that compared concatenated brand and generic names against `ePillID`'s `pilltype_id`/label strings. These steps materially improved metadata availability without inflating duplicates, allowing the Decision Tree to train on a better-populated feature space.

### Class imbalance and evaluation

The class distribution was highly skewed toward "other," which encouraged the Decision Tree to over-predict that class and depressed minority recalls. We mitigated this by enforcing stratified splits, using `class_weight="balanced"`, and pruning ultra-rare classes within each training subset prior to fitting. Because Accuracy can mask minority underperformance in such settings, we emphasized Macro-F1 alongside Accuracy in all summaries. These changes yielded a more stable Decision Tree and a fairer read on progress across all classes.

### Text sparsity and enrichment strategy

A significant fraction of joined rows lacked usable label text fields. To prevent the text-only DT from collapsing to "other," we trained text models only on rows with non-empty text and removed ultra-rare labels within that text subset. Importantly, our openFDA enrichment intentionally sampled 5,000 SPL set IDs out of 121,677 total (seeded for reproducibility) to stay within time and rate-limit budgets. We caught API responses, normalized and truncated text to cap extremes, and then merged the results back into the tidy FDALabel table. This selective enrichment was sufficient for strong text modeling while remaining efficient and reproducible.

### Controlling DT overfitting

High-cardinality one-hot features from `route`/`dosage` created overfitting pressure in the Decision Tree. We constrained complexity via `max_depth=10` and `min_samples_leaf=20` and performed rare-class pruning before training. These choices produced interpretable splits and prevented the Decision Tree from simply memorizing narrow categories, raising Macro-F1 relative to an unconstrained baseline.

### Handling prediction uncertainty

Single argmax labels hid uncertainty in ambiguous cases. We implemented a top-k until 80% cumulative probability routine on the DT's predicted probabilities to expose a compact, probability-mass-covering set of candidate labels. This mechanism better reflects confidence at inference time and is practical for downstream human-in-the-loop verification.

### Feature engineering (DT)

For the Decision Tree, we concatenated "indications," "warnings," "adverse," "brand_name", and "generic_name" into a normalized text source, then derived lightweight keyword indicators (e.g., "hypertension," "statin," "insulin," "analgesic," "antibiotic," "asthma," "omeprazole"). Categorical metadata "route" and "dosage_form was one-hot encoded. This compact feature set balanced interpretability with coverage, enabling a resilient baseline even when full text was unavailable.

### Results

TF-IDF + Logistic Regression achieved 0.99 (99%) Accuracy and 0.90 (90%) Macro-F1, with confusion matrices

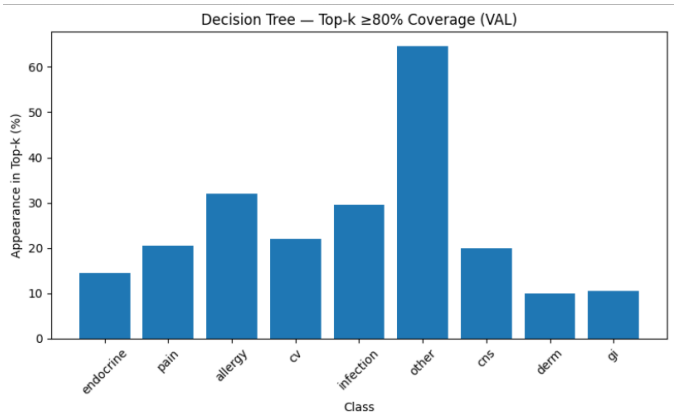


Fig. 15. DT top-k coverage per class using an 80% cumulative-probability rule.

showing strong diagonals and only minor bleed between "derm" and "other."

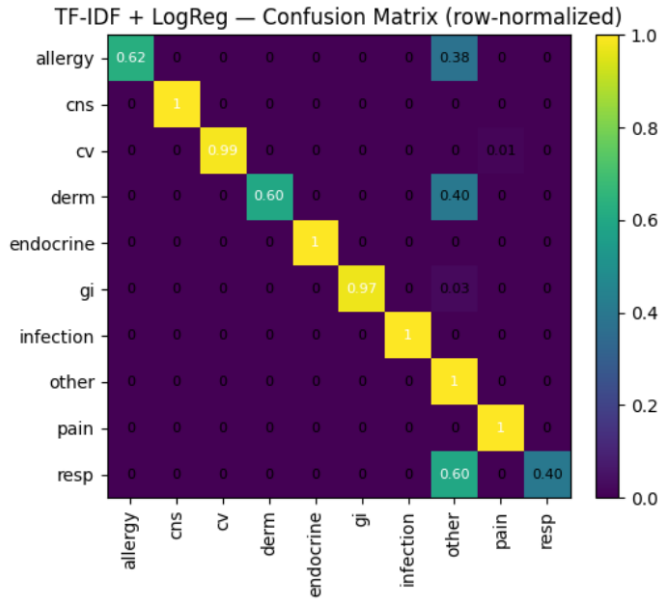


Fig. 16. Row-normalized confusion matrix for TF-IDF+LogReg on the text subset (Acc0.99, Macro-F10.90)

The Decision Tree (tubular, all rows) reached 0.67 (67%) Accuracy and 0.34 (34%) Macro-F1, a meaningful baseline given the imbalance and partial metadatav

**Model summary**

The largest gains came from:

- Using text where available via TF-IDF bi-grams with class-balanced Logistic Regression.
- Reducing label noise through rare-class pruning within each training subset.
- Constraining tree complexity to prevent memorization.
- Improving join coverage with richest-text collapsing and fuzzy matching.

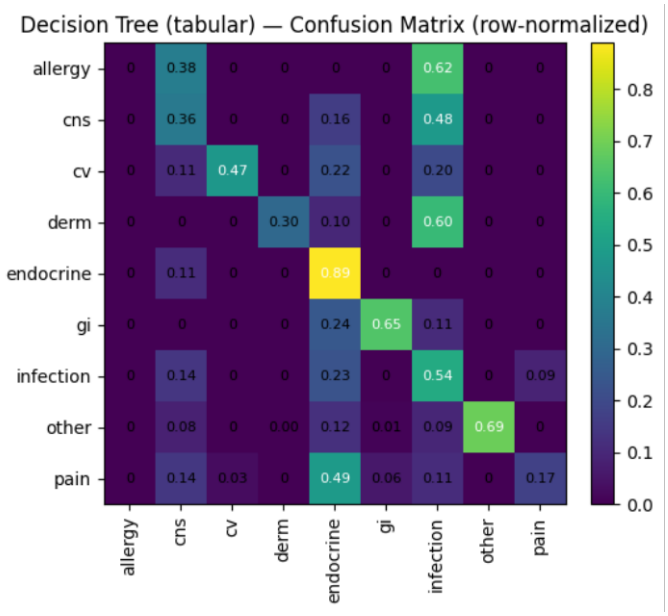


Fig. 17. Row-normalized confusion matrix for the tabular DT baseline (Acc0.67, Macro-F10.34).

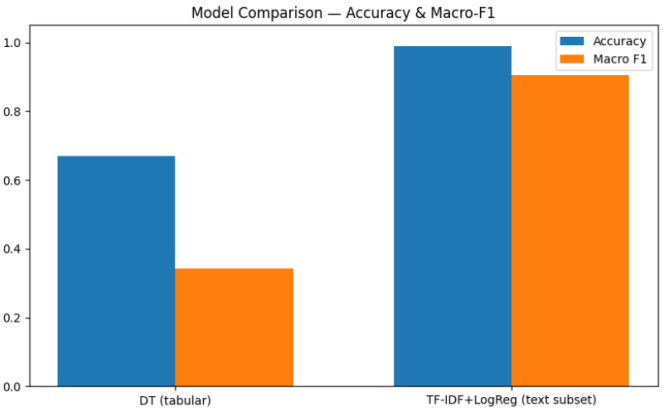


Fig. 18. Comparison across TF-IDF+LogReg (text subset), DT (tabular), and DT (text-only)

- Judging with Macro-F1, made the impact on minority classes visible and actionable.

## IV. Comparison

**CNN vs. SVM vs. Decision Tree**

We evaluated four models on an 80/20 stratified split: a CNN (EfficientNet-B0) on images, an SVM with PCA features, a Decision Tree (DT) on tabular + keyword signals, and a TF-IDF+LogReg text model on FDALabel fields. At convergence they achieved: CNN 0.649 Accuracy / 0.604 F1 (weighted F1 from the training log), SVM 0.298 / 0.274, DT 0.670 / 0.343 (macro-F1), and TF-IDF+LogReg 0.989 / 0.905 (macro-F1). Despite the F1 definition mismatch (weighted for CNN vs. macro for others), the ranking is clear: text-first TF-

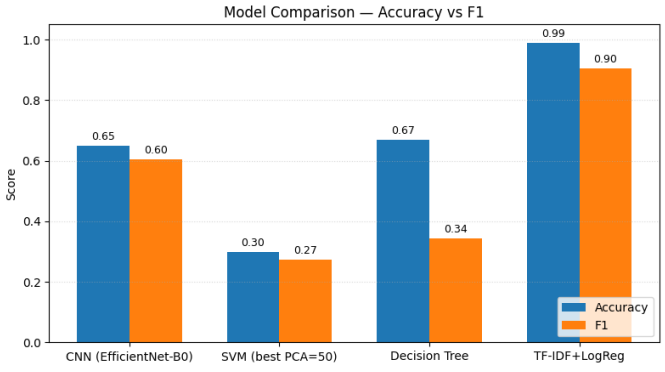


Fig. 19. Accuracy vs. F1 across all ML algorithms

IDF+LogReg dominates; DT is a reasonable accuracy baseline but weak on minority classes; CNN is mid-pack; SVM trails.

Why this pattern? The condition buckets are fundamentally lexical (e.g., "hypertension," "antibiotic," "insulin"), so our cleaned/enriched FDALabel text sampled 5,000 of 121,677 SPL set IDs from openFDA and normalized into bigrams gives TF-IDF+LogReg a linearly separable signal, driving its very high macro-F1 across classes. The DT benefits from interpretable route/dosage one-hots and keyword flags, but limited ePillIDFDALabel join coverage ( 39.5%) and high-cardinality one-hots bias it toward the majority "other" class; accuracy looks okay while macro-F1 drops even with class weighting and depth/leaf constraints. The CNN, trained only on pill appearance, must infer therapeutic use indirectly via the join, and visual cues are weak proxies for "condition," so it plateaus around 0.65/0.60. The SVM underperforms because PCA-compressed features over low-shot, look-alike pills offer poor separability without richer learned embeddings.

**Comparison to existing work.**

Usuyama et al. (CVPR 2020, ePillID) target fine-grained pill identity under low-shot, real-world imaging using a CNN encoder with a bilinear head and multi-head metric learning, evaluated on identity metrics (e.g., top-k ID). Our objective is different—mapping pills to therapeutic conditions via FDALabel after an ePillID→NDC join—so it is expected that TF-IDF+LogReg excels on condition labels (Acc 0.989, Macro-F1 0.905), while the image-only CNN sits mid-pack (Acc 0.649, F1 0.604). The two are complementary: a metric-learning ID stage like Usuyama's could strengthen our identity step, after which our text model (or DT fallback) provides the condition mapping.

Kavitha et al. (2024) focus on detection + identity with a YOLO-based detector, MobileNet features, and custom classification over 20 classes, using standard resizing/augmentation for generalization. That line emphasizes real-time localization and type recognition, not condition inference. In practice, their detector could front-end our system (crop/track pills), while our text-driven classifier supplies the usage/condition output that object detectors do not address.

For condition mapping, text supervision wins. DT remains a transparent metadata baseline and CNN/SVM are better reserved for the identity stage or as components in a future late-fusion pipeline.

## V. Future Directions

Given additional time and resources, to enhance the performance of the CNN for pill identification we could have explored more complex architectures in the EfficientNet family. Using class-aware loss functions such as focal loss or label smoothing could improve performance or rare classes. Finally, creating a checkpoint for early stopping would allow for better efficient training and prevent overfitting. This will ultimately improve validation, accuracy, and reliability in real-world application.

In terms of handling the classes with only 2 images, the best course of action to improve the runtime would be to add more images to these classes or choose a dataset with more images. Ideally, all classes should have more than 5 images and all classes can be classified via the CNN algorithm. Another strategy for dealing with the scarce dataset to explore would be artificially augmenting the data to create more training examples allowing for a testing and training data split as opposed to using cross-validation. Another strategy moving forward would be to explore other training strategies learned later on in the course such as using unsupervised learning techniques.

To push beyond the current text-driven ceiling (0.99/0.90) and lift the DT baseline (0.67/0.34), we'd expand enrichment beyond the sampled 5,000 of 121,677 SPL set IDs via automated openFDA/DailyMed pulls and stricter NDC/name normalization to raise join rate. On modeling, we'd replace high-cardinality one-hots with target/frequency encoding and benchmark CatBoost/LightGBM/XGBoost (with class weights and monotonic constraints), plus a hierarchical "other vs. non-other" stage before fine classes. We'd calibrate probabilities (Platt/isotonic) and add conformal prediction so top-k has measurable coverage. Finally, we'd explore late-fusion of a CNN pill-image embedding with FDALabel features, use active learning on uncertain/rare classes, and apply Bayesian hyperparameter search + SHAP to guide feature pruning and interpretability.

## VI. Overall Conclusion

Overall, this project demonstrated that text-first approaches specifically TF-IDF combined with Logistic Regression performed best, achieving strong results with an accuracy of 0.989 and a macro-F1 of 0.905. In contrast, image-only models such as CNNs and SVMs lagged behind, highlighting the challenges of pill identification using visual features alone. While the current solution provides a solid baseline, it is clear that the problem is not yet fully solved. Finally, future work should explore larger and more diverse datasets, improved pill-image embeddings, and the integration of FDALabel metadata to better leverage both text and image

like Transformer-based models such as BERT.

## REFERENCES


[1] Usuyama, N., Delgado, N. L., Hall, A. K., & Lundin, J. (2020). ePillID Dataset: A Low-Shot Fine-Grained Benchmark for Pill Identification. In *CVPR Workshops*.

[2] Ponte, E., Amparo, X., Huang, K., & Kwak, D. (2023). Automatic Pill Identification System based on Deep Learning and Image Preprocessing. In *CSCE 2023*. IEEE. https://doi.org/10.1109/CSCE60160.2023.00324

[3] Lee, S., Kim, H., & Park, J. (2023). An accurate deep learning–based system for automatic pill identification: model development and validation. *Journal of Medical Internet Research*, 25(1), e41043.

[4] Kavitha, N., Stefi, A., Varsha, P. M., Sumana, S., & Simha, V. V. (2024). Using machine learning models to detect and identify pills. *Indiana Journal of Multidisciplinary Research*, 4(3), 155–161.

[5] Simplilearn, "Convolutional Neural Network Tutorial," simplilearn.com. Accessed: Aug. 3, 2025.

[6] Datacamp, "Introduction to CNNs," datacamp.com. Accessed: Aug. 3, 2025.

[7] IBM, "Support Vector Machine (SVM)," ibm.com. Accessed: Aug. 3, 2025.

[8] scikit-learn, "Support Vector Machines," scikit-learn.org. Accessed: Aug. 3, 2025.

[9] Analytics Vidhya, "Image Classification Using SVM," medium.com. Accessed: Aug. 3, 2025.

[10] scikit-learn, "Decision Trees," scikit-learn.org. Accessed: Aug. 3, 2025.

[11] Fiveable, "Decision Trees for Image Analysis," fiveable.me. Accessed: Aug. 3, 2025.