

Image Based Medication Identification using C3PI

Andrew Rodriguez

Introduction:

I picked this project because it merges computer vision with something actually useful in the real world: identifying pills from photos. It's an interesting problem that I wanted to tackle. Today, medications are imprinted with identification numbers which, paired with pill size, color, shape, etc can be used to identify a pill. A model that can just look at a pill and tell you what it is seemed like a fun challenge. If you can get reliable predictions from just a photo this could lend towards something larger and have an impact. A good predictive model for pill identification could have several important applications. For people with visual impairments or reading difficulties, being able to identify medications through a photo could be way more accessible than trying to read tiny imprint codes. The model could also be integrated into robotic systems for automated medication sorting or inventory management in healthcare settings.

The dataset is large with a lot of potential: over 130,000 images with varied conditions. Getting something to work across that many different classes while still generalizing well would be a solid proof of concept. With this much data and varied conditions, it's obvious that this will require a large amount of compute power and time. This project merely takes a small step into this effort and it is clear there's much more to be done.

Methods:

Data Setup

I used PySpark for distributed processing since the dataset was large. Set up a Spark session with 32GB driver memory, 120GB executor memory, and had to adjust configurations on the fly to resolve memory and timeout issues that came up during processing.

Data Exploration

Started with the C3PI dataset's reference images from table.csv which was about 4,332 entries with 2,111 unique drug classes. Each entry had multiple image resolutions available (120px to original). I selected the 300px resolution as a middle ground between performance and efficiency as the original resolution and 600px resolution resulted in memory and stability issues.

Furthermore, the reference dataset was too small for decent performance, so I downloaded the consumer-grade images (about 12,000 additional images across 338 drug classes). To attempt at training on realistic data.

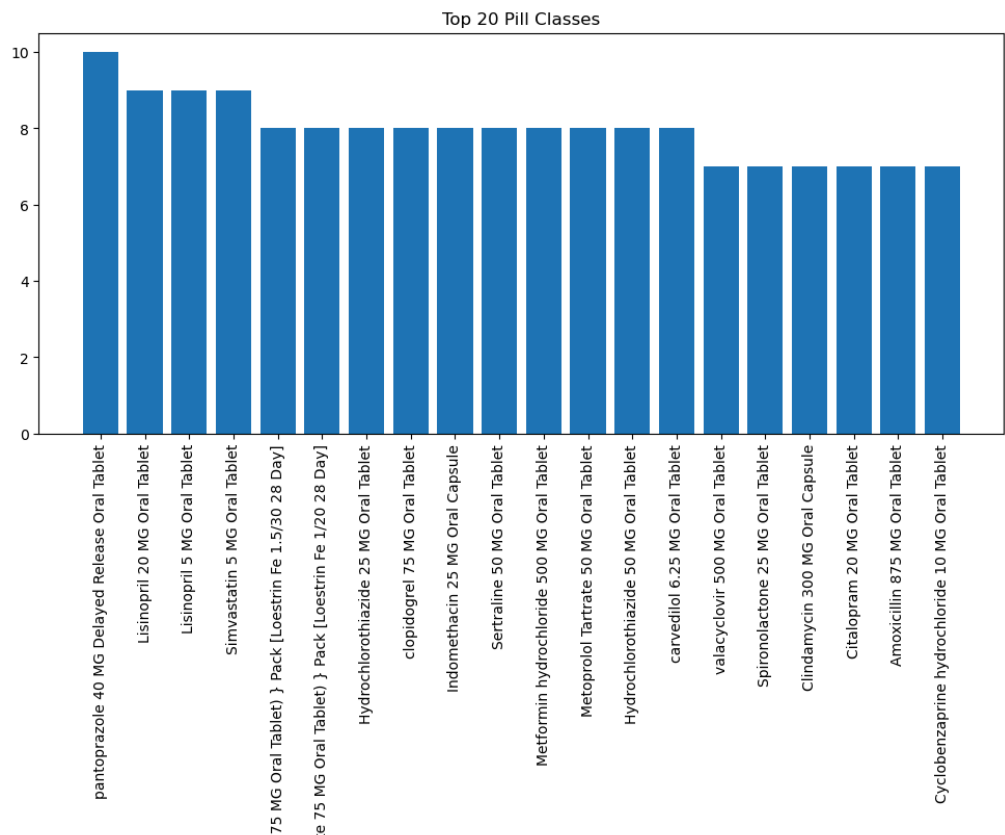


Figure 1: Top 20 Pill Classes




	NDC: 00093-0150-01 (FRONT)		IMPRINT: 3,TV,150
	SIZE: 10mm	SCORE: 1	COLOR: WHITE
	COATING: N/A	SYMBOL: FALSE	SHAPE: ROUND
	Office of High Performance Computing & Communications		Computational Photography Project for Pill Identification (C3PI)
	CS: sRGB v1.31 (Canon)	DPI: 2024	IMAGING: Medicos Consultants LLC, 16th June 2015

Figure 2: Consumer Grade Image Sample



Figure 3: Reference Image Sample

Data Preprocessing

Before training, I had to process the images into something a model could use. Here's what I did:

- Resized all images to 224x224 using Lanczos resampling
- Converted all images to RGB
- Normalized pixel values to the 0–1 range
- Flattened the images into 1D vectors
- Wrote a PySpark UDF so I could run all of this in parallel on the cluster
- Saved data into parquet file for input into Spark

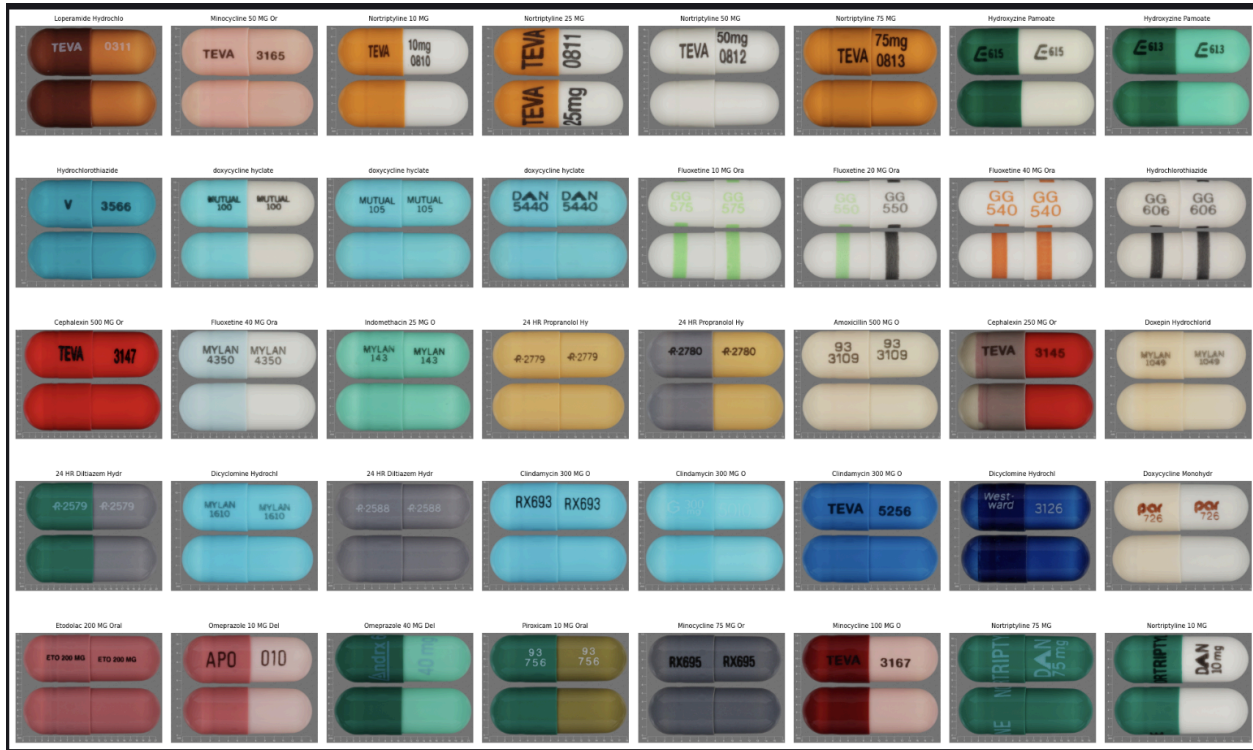


Figure 4: Reference Images

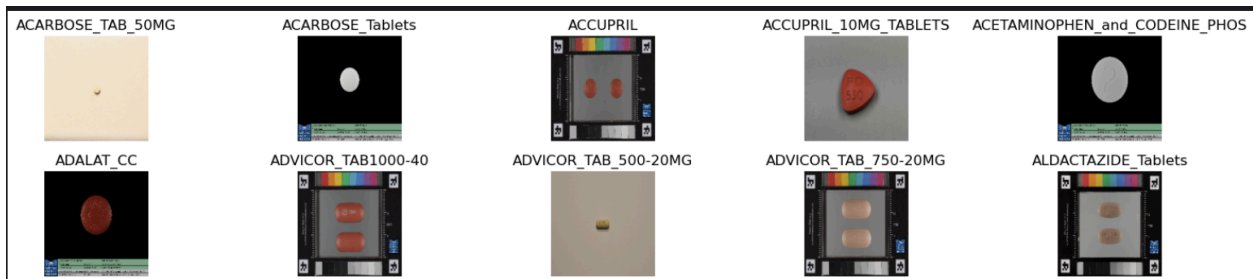


Figure 5: Consumer Images Samples

Model 1: Random Forest

Once the data was ready, I trained a Random Forest classifier using Spark. I filtered the data to just the top 20 most common drug names to reduce class imbalance and keep the training manageable.

Random Forest Settings:

- numTrees = 30
- maxDepth = 15
- subsamplingRate = 0.8

Model 2: Logistic Regression

Same setup as above, just swapped the model out for logistic regression. This one ran faster and gave a decent baseline to compare with.

Logistic Regression Settings:

- maxIter = 5
- regParam = 0.1
- elasticNetParam = 0.1
- Threshold = 0.5

Model 3: ResNet18 CNN

This was more of a sanity check using the 12k consumer pill images. I loaded everything into a PyTorch ImageFolder dataset and ran a basic ResNet18 from torchvision.models. No pretrained weights and only trained for one epoch on CPU to see if it even worked.

Setup:

- InputSize = 128x128
- batchSize = 32
- optimizer = adam
- loss = crossentropy

Results:

Model 1: Random Forest

This model was trained on the original RxImage reference data using just the top 20 most common drug names. I used flattened 224x224 image vectors as features.

- Accuracy: 0.31
- F1 Score: 0.29
- Weighted Precision: 0.32
- Weighted Recall: 0.31

Model 2: Logistic Regression

- Accuracy: 0.28
- F1 Score: 0.32
- Weighted Precision: 0.44

- Weighted Recall: 0.28

Model 3: ResNet18 CNN

- Accuracy: 0.0264
- F1 Score: 0.0057

Discussion:

The C3PI dataset was quite large, with high quality reference images and a much larger set of consumer images. I initially used the reference set, which contained around 4,000 entries. However, this set was too heavily imbalanced. There were more than 2,000 drug classes, but many of them only had one or two images. This made it difficult to train any model with meaningful generalization. To address this, I switched to the consumer-grade image collection, which offered more data. While the larger size helped, the image quality was highly variable. Some photos were blurry, others had poor lighting, and many had inconsistent backgrounds. These issues made preprocessing difficult and introduced additional noise into the dataset resulting in the unusable CNN model.

I settled on using 300-pixel resolution images as a compromise between image clarity and processing limits. Attempts to use higher resolutions led to repeated out of memory errors with Spark.

Performance Issues:

Initially, I started work on my local machine with 64gb high performance memory but experienced out of memory issues so I resorted to leveraging SDSC. Despite using 200gb memory, Spark required constant tuning. I had to adjust memory allocation for both executors and the driver, change partition settings, and resolve issues related to how Spark handles column names with spaces. These problems consumed much of the time and made the overall workflow fragile and prone to failure. I would train a model for many hours just to return to an error.

Random Forest and Logistic Regression Performance:

The first two models I trained were Random Forest and Logistic Regression. To reduce complexity, I trained both models on the top 20 most frequent drug classes in the dataset. This helped reduce class imbalance and allowed the models to run within resource limits.

Random Forest achieved about 31 percent accuracy with an F1 score of 0.29. Logistic Regression performed slightly worse in terms of accuracy, around 28 percent, but had a higher weighted precision. This suggests that Logistic Regression was more confident in the predictions it made, though those predictions were not necessarily more accurate overall.

With that said, the performance was very bad.

ResNet18 CNN Performance:

To test whether a deep learning model could better handle this task, I ran a basic ResNet18 model in PyTorch. The model was trained on the consumer images for one epoch without pretrained weights. It was run entirely on CPU. The results were very bad, with an accuracy of 2.6 percent and a weighted F1 score that was practically zero. This test was not expected to perform well but was useful to verify that a convolutional neural network could be integrated with the dataset.

Conclusion:

This project was an attempt to build a system that can identify pills from images using machine learning. While the results were limited, the effort provided valuable insight into the challenges of working with large messy datasets and the technical constraints of distributed computing.

The biggest takeaway is that the main bottleneck in this kind of project is not the model architecture but the data. If I were to attempt this project again, I would focus almost entirely on cleaning and curating the consumer image dataset. A convolutional neural network would likely be the best modeling approach, but it would only be effective if the input data is consistent and clean. Based on what I experienced, at least 95 percent of the effort would need to go into data preparation and organization before model training even begins.

This was also my first time working with the SDSC cluster and Spark at this scale. I was unfamiliar with the infrastructure at the start and ran into a number of problems with memory limits and long-running jobs going to waste. Spark required more fine-tuning than I expected. Despite the difficulties, I learned a lot from troubleshooting and figuring out how to get things running.

If I had the opportunity to do this again, I know what not to do. Now that I have a better understanding of both the technical tools and the nature of the data, I believe I could approach the problem with a more focused and experienced plan. While the accuracy scores were not high, the experience was valuable and laid a solid foundation for myself to attempt more projects in image based modeling.

Statement of Collaboration:

Not applicable. Solo team.